

# Nachschlagewerk fuer die Programmiersprache MQL5

fuer Kundenterminal MetaTrader 5

**Lernen Sie die Sprache MQL5 und  
loesen Sie jegliche Aufgaben:**

- Erzeugung von eigenen Indikatoren fuer technische Analyse unterschiedlicher Kompliziertheit
- Autotrading - Automatisieren eines Handelssystems fuer Arbeit in verschiedenen Finanzmaerkten
- Ausarbeitung von analytischen Instrumenten auf Grundlage mathematischer Erkenntnisse und klassischer Methoden
- Schreiben von handelsinformationellen Systemen fuer ein grosses Komplex der Aufgaben (Trading, Monitoring, Signale usw)

# Inhalt

## Nachschlagewerk MQL5

50

|                                                      |            |
|------------------------------------------------------|------------|
| <b>1 Grundlagen der Sprache.....</b>                 | <b>52</b>  |
| <b>Syntax .....</b>                                  | <b>53</b>  |
| Kommentare.....                                      | 54         |
| Identifikatoren.....                                 | 55         |
| Reservierte Wörter .....                             | 56         |
| <b>Datentypen .....</b>                              | <b>58</b>  |
| Ganzzahlige Typen.....                               | 59         |
| Typen char, short, int und long.....                 | 60         |
| Symbolkonstanten.....                                | 63         |
| Typ datetime.....                                    | 66         |
| Typ color .....                                      | 67         |
| Typ bool .....                                       | 68         |
| Aufzählungen.....                                    | 69         |
| Realtypen (double, float).....                       | 71         |
| Typ string .....                                     | 76         |
| Strukturen und Klassen .....                         | 77         |
| Objekt des dynamischen Feldes .....                  | 88         |
| Typenreduzierung .....                               | 90         |
| Typ void und Konstante NULL.....                     | 96         |
| Objektanzeiger .....                                 | 97         |
| Referenzen, Modifikator& und Schlüsselwort this..... | 98         |
| <b>Operationen und Ausdrücke .....</b>               | <b>100</b> |
| Ausdrücke.....                                       | 101        |
| Arithmetische Operationen.....                       | 102        |
| Zuordnungsoperationen.....                           | 103        |
| Vergleichsoperationen .....                          | 104        |
| Logische Operationen.....                            | 105        |
| Bitorganisierte Operationen.....                     | 107        |
| Andere Operationen .....                             | 110        |
| Prioritäten und Operationsordnung.....               | 114        |
| <b>Operatoren .....</b>                              | <b>116</b> |
| Zusammengesetzter Operator.....                      | 118        |
| Operator-Ausdruck .....                              | 119        |
| Rücksetzoperator return.....                         | 120        |
| Konditionaloperator if-else .....                    | 121        |
| Ternärer Operator ?:.....                            | 122        |
| Umschalteroperator switch .....                      | 124        |
| Loop-Operator while.....                             | 126        |
| Loop-Operator for .....                              | 127        |
| Loop-Operator do while .....                         | 129        |
| Endoperator break.....                               | 130        |
| Fortsetzungsoperator continue.....                   | 131        |
| Operator der Objekterzeugung new.....                | 132        |
| Operator der Objektenfernung delete .....            | 134        |
| <b>Funktionen .....</b>                              | <b>135</b> |
| Funktionsaufruf.....                                 | 137        |
| Parameterübertragung.....                            | 138        |
| Überladen von Funktionen.....                        | 141        |
| Überladen von Operatoren .....                       | 144        |
| Beschreibung der Aussenfunktionen.....               | 156        |
| Exportieren der Funktionen.....                      | 158        |
| Ereignisbearbeiter .....                             | 159        |

|                                                                |            |
|----------------------------------------------------------------|------------|
| <b>Variablen</b> .....                                         | <b>171</b> |
| Lokale Variablen.....                                          | 174        |
| Formale Parameter.....                                         | 176        |
| Statische Parameter.....                                       | 178        |
| Globale Variablen.....                                         | 180        |
| Input Variablen.....                                           | 181        |
| Extern Variablen.....                                          | 185        |
| Initialisierung der Variablen.....                             | 186        |
| Sichtbereich und Lebensdauer der Variablen.....                | 188        |
| Erzeugung und Entfernung der Objekte.....                      | 190        |
| <b>Preprozessor</b> .....                                      | <b>193</b> |
| Makrosubstitution (#define).....                               | 194        |
| Programmeigenschaften (#property).....                         | 197        |
| Include Dateien (#include).....                                | 202        |
| Import der Funktionen (#import).....                           | 203        |
| Условная компиляция (#ifdef, #ifndef, #else, #endif).....      | 205        |
| <b>Objektorientiertes Programmieren</b> .....                  | <b>207</b> |
| Inkapsulation und Erweiterungsfähigkeit der Typen.....         | 209        |
| Vererbung.....                                                 | 212        |
| Polymorphismus.....                                            | 217        |
| Überladen.....                                                 | 221        |
| Virtuelle Funktionen.....                                      | 222        |
| Statische Klassenmitglieder.....                               | 225        |
| Funktionsschablonen.....                                       | 229        |
| <b>2 Standardkonstanten, Aufzählungen und Strukturen</b> ..... | <b>232</b> |
| <b>Chartkonstanten</b> .....                                   | <b>233</b> |
| Typen der Chartereignisse.....                                 | 234        |
| Chartperioden.....                                             | 241        |
| Charteigenschaften.....                                        | 243        |
| Positionieren des Charts.....                                  | 249        |
| Darstellung des Charts.....                                    | 250        |
| Beispiele von Arbeiten mit dem Chart.....                      | 252        |
| <b>Objektkonstanten</b> .....                                  | <b>310</b> |
| Objekttypen.....                                               | 311        |
| OBJ_VLINE .....                                                | 313        |
| OBJ_HLINE .....                                                | 318        |
| OBJ_TREND .....                                                | 323        |
| OBJ_TRENDBYANGLE .....                                         | 330        |
| OBJ_CYCLES.....                                                | 336        |
| OBJ_ARROWED_LINE.....                                          | 342        |
| OBJ_CHANNEL.....                                               | 348        |
| OBJ_STDDEVCHANNEL.....                                         | 355        |
| OBJ_REGRESSION.....                                            | 362        |
| OBJ_PITCHFORK.....                                             | 368        |
| OBJ_GANNLIN.....                                               | 376        |
| OBJ_GANNFAN.....                                               | 383        |
| OBJ_GANNGRID.....                                              | 390        |
| OBJ_FIBO .....                                                 | 397        |
| OBJ_FIBOTIMES.....                                             | 404        |
| OBJ_FIBOFAN.....                                               | 411        |
| OBJ_FIBOARC.....                                               | 418        |
| OBJ_FIBOCHANNEL.....                                           | 425        |
| OBJ_EXPANSION.....                                             | 433        |
| OBJ_ELLIOTWAVE5.....                                           | 441        |
| OBJ_ELLIOTWAVE3.....                                           | 449        |
| OBJ_RECTANGLE .....                                            | 456        |
| OBJ_TRIANGLE.....                                              | 462        |
| OBJ_ELLIPSE.....                                               | 469        |

|                                                          |            |
|----------------------------------------------------------|------------|
| OBJ_ARROW_THUMB_UP.....                                  | 475        |
| OBJ_ARROW_THUMB_DOWN.....                                | 481        |
| OBJ_ARROW_UP.....                                        | 487        |
| OBJ_ARROW_DOWN.....                                      | 493        |
| OBJ_ARROW_STOP.....                                      | 499        |
| OBJ_ARROW_CHECK.....                                     | 505        |
| OBJ_ARROW_LEFT_PRICE.....                                | 511        |
| OBJ_ARROW_RIGHT_PRICE.....                               | 516        |
| OBJ_ARROW_BUY.....                                       | 521        |
| OBJ_ARROW_SELL.....                                      | 526        |
| OBJ_ARROW.....                                           | 531        |
| OBJ_TEXT.....                                            | 537        |
| OBJ_LABEL.....                                           | 543        |
| OBJ_BUTTON.....                                          | 551        |
| OBJ_CHART.....                                           | 558        |
| OBJ_BITMAP.....                                          | 565        |
| OBJ_BITMAP_LABEL.....                                    | 572        |
| OBJ_EDIT.....                                            | 579        |
| OBJ_EVENT.....                                           | 586        |
| OBJ_RECTANGLE_LABEL.....                                 | 591        |
| Objekteigenschaften.....                                 | 597        |
| Methode der Objektbindung.....                           | 604        |
| Bindungswinkel.....                                      | 608        |
| Objektsichtbarkeit.....                                  | 610        |
| Levels von Elliott Wellen.....                           | 613        |
| Gann Objekte.....                                        | 614        |
| Web-Farben.....                                          | 616        |
| Wingdings.....                                           | 618        |
| <b>Indikatorkonstanten.....</b>                          | <b>619</b> |
| Preiskonstanten.....                                     | 620        |
| Glaettungsmethoden.....                                  | 623        |
| Linien der Indikatoren.....                              | 624        |
| Zeichnungsstile.....                                     | 626        |
| Eigenschaften der Benutzerobjekte.....                   | 630        |
| Typen der Indikatoren.....                               | 633        |
| Identifikatoren der Datentypen.....                      | 635        |
| <b>Medium Zustand.....</b>                               | <b>636</b> |
| Zustand des Kundenterminals.....                         | 637        |
| Information über das ausgeführte MQL5-Programm.....      | 640        |
| Information über das Symbol.....                         | 643        |
| Information über das Konto.....                          | 656        |
| Teststatistik.....                                       | 660        |
| <b>Handelskonstanten.....</b>                            | <b>664</b> |
| Information über historische Daten des Instrumentes..... | 665        |
| Ordereigenschaften.....                                  | 666        |
| Eigenschaften der Positionen.....                        | 670        |
| Eigenschaften der Deals.....                             | 672        |
| Typen der Handelsoperationen.....                        | 675        |
| Types of Trade Transactions.....                         | 676        |
| Handelsordern in DOM.....                                | 679        |
| Signal Properties.....                                   | 680        |
| <b>Benannte Konstanten.....</b>                          | <b>682</b> |
| Vorbestimmte Makrosubstitutionen.....                    | 683        |
| Mathematische Konstanten.....                            | 685        |
| Konstanten der numerischen Typen.....                    | 687        |
| Deinitialisierungsgruende.....                           | 690        |
| Pruefung des Objektanzeigers.....                        | 692        |
| Andere Konstanten.....                                   | 693        |



|                                                            |            |
|------------------------------------------------------------|------------|
| <b>Datenstrukturen</b>                                     | <b>697</b> |
| Datenstruktur                                              | 698        |
| Struktur der Eingabeparameter des Anzeigers                | 699        |
| Struktur der historischen Daten                            | 700        |
| DOM Struktur                                               | 701        |
| Struktur der Handelsanforderung                            | 702        |
| Struktur der Ergebnisse der Prüfung der Handelsanforderung | 706        |
| Struktur des Ergebnisses der Handelsanforderung            | 707        |
| Structure of a Trade Transaction                           | 711        |
| Struktur fuer Erfassung der laufenden Preise               | 715        |
| <b>Kodes der Fehler und Warnungen</b>                      | <b>716</b> |
| Rückgabecodes des Handelsservers                           | 717        |
| Compiler Warnungen                                         | 719        |
| Kompilierungsfehler                                        | 723        |
| Ausführungsfehler                                          | 735        |
| <b>Eingabe/Ausgabe Konstanten</b>                          | <b>745</b> |
| Flaggen der Dateieröffnung                                 | 746        |
| Dateieigenschaften                                         | 749        |
| Positionieren innerhalb der Dateien                        | 751        |
| Kodeseite Verwenden                                        | 752        |
| MessageBox                                                 | 753        |
| <b>3 MQL5 Programme</b>                                    | <b>755</b> |
| Programmausführung                                         | 756        |
| Handelserlaubnis                                           | 763        |
| Ereignisse des Kundenterminals                             | 767        |
| Ressourcen                                                 | 771        |
| Aufruf der Importfunktionen                                | 780        |
| Ausführungsfehler                                          | 782        |
| Testen von Handelsstrategien                               | 783        |
| <b>4 Zugeordnete Konstanten</b>                            | <b>809</b> |
| _Digits                                                    | 810        |
| _Point                                                     | 811        |
| _LastError                                                 | 812        |
| _Period                                                    | 813        |
| _RandomSeed                                                | 814        |
| _StopFlag                                                  | 815        |
| _Symbol                                                    | 816        |
| _UninitReason                                              | 817        |
| <b>5 Allgemeine Funktionen</b>                             | <b>818</b> |
| Alert                                                      | 820        |
| CheckPointer                                               | 821        |
| Comment                                                    | 822        |
| CryptEncode                                                | 824        |
| CryptDecode                                                | 826        |
| DebugBreak                                                 | 827        |
| ExpertRemove                                               | 828        |
| GetPointer                                                 | 830        |
| GetTickCount                                               | 833        |
| MessageBox                                                 | 834        |
| PeriodSeconds                                              | 835        |
| PlaySound                                                  | 836        |
| Print                                                      | 837        |
| PrintFormat                                                | 839        |
| ResetLastError                                             | 845        |
| ResourceCreate                                             | 846        |
| ResourceFree                                               | 848        |
| ResourceReadImage                                          | 849        |

|                                         |            |
|-----------------------------------------|------------|
| ResourceSave .....                      | 850        |
| SetUserError .....                      | 851        |
| SendFTP .....                           | 852        |
| SendMail .....                          | 853        |
| SendNotification .....                  | 854        |
| Sleep .....                             | 855        |
| TerminalClose .....                     | 856        |
| TesterStatistics .....                  | 858        |
| TesterWithdrawal .....                  | 859        |
| WebRequest .....                        | 860        |
| ZeroMemory .....                        | 866        |
| <b>6 Operationen mit Arrays .....</b>   | <b>867</b> |
| ArrayBsearch .....                      | 868        |
| ArrayCopy .....                         | 872        |
| ArrayCompare .....                      | 877        |
| ArrayFree .....                         | 878        |
| ArrayGetAsSeries .....                  | 887        |
| ArrayInitialize .....                   | 890        |
| ArrayFill .....                         | 892        |
| ArrayIsDynamic .....                    | 894        |
| ArrayIsSeries .....                     | 897        |
| ArrayMaximum .....                      | 899        |
| ArrayMinimum .....                      | 910        |
| ArrayRange .....                        | 921        |
| ArrayResize .....                       | 922        |
| ArraySetAsSeries .....                  | 925        |
| ArraySize .....                         | 927        |
| ArraySort .....                         | 929        |
| <b>7 Datenverarbeitung .....</b>        | <b>934</b> |
| CharToString .....                      | 936        |
| CharArrayToString .....                 | 937        |
| ColorToARGB .....                       | 938        |
| ColorToString .....                     | 940        |
| DoubleToString .....                    | 941        |
| EnumToString .....                      | 942        |
| IntegerToString .....                   | 944        |
| ShortToString .....                     | 945        |
| ShortArrayToString .....                | 946        |
| TimeToString .....                      | 947        |
| NormalizeDouble .....                   | 948        |
| StringToCharArray .....                 | 950        |
| StringToColor .....                     | 951        |
| StringToDouble .....                    | 952        |
| StringToInteger .....                   | 953        |
| StringToShortArray .....                | 954        |
| StringToTime .....                      | 955        |
| StringFormat .....                      | 956        |
| <b>8 Mathematische Funktionen .....</b> | <b>957</b> |
| MathAbs .....                           | 958        |
| MathArccos .....                        | 959        |
| MathArcsin .....                        | 960        |
| MathArctan .....                        | 961        |
| MathCeil .....                          | 962        |
| MathCos .....                           | 963        |
| MathExp .....                           | 964        |
| MathFloor .....                         | 965        |
| MathLog .....                           | 966        |

|                                 |             |
|---------------------------------|-------------|
| MathLog10 .....                 | 967         |
| MathMax .....                   | 968         |
| MathMin .....                   | 969         |
| MathMod .....                   | 970         |
| MathPow .....                   | 971         |
| MathRand .....                  | 972         |
| MathRound .....                 | 973         |
| MathSin .....                   | 974         |
| MathSqrt .....                  | 975         |
| MathSrand .....                 | 976         |
| MathTan .....                   | 979         |
| MathIsValidNumber .....         | 980         |
| <b>9 Stringfunktionen .....</b> | <b>981</b>  |
| StringAdd .....                 | 983         |
| StringBufferLen .....           | 985         |
| StringCompare .....             | 986         |
| StringConcatenate .....         | 988         |
| StringFill .....                | 989         |
| StringFind .....                | 990         |
| StringGetCharacter .....        | 991         |
| StringInit .....                | 992         |
| StringLen .....                 | 993         |
| StringReplace .....             | 994         |
| StringSetCharacter .....        | 995         |
| StringSplit .....               | 997         |
| StringSubstr .....              | 998         |
| StringToLower .....             | 999         |
| StringToUpper .....             | 1000        |
| StringTrimLeft .....            | 1001        |
| StringTrimRight .....           | 1002        |
| <b>10 Datum und Zeit.....</b>   | <b>1003</b> |
| TimeCurrent .....               | 1004        |
| TimeTradeServer .....           | 1005        |
| TimeLocal .....                 | 1006        |
| TimeGMT .....                   | 1007        |
| TimeDaylightSavings .....       | 1008        |
| TimeGMTOffset .....             | 1009        |
| TimeToStruct .....              | 1010        |
| StructToTime .....              | 1011        |
| <b>11 Kontoinformation.....</b> | <b>1012</b> |
| AccountInfoDouble .....         | 1013        |
| AccountInfoInteger .....        | 1014        |
| AccountInfoString .....         | 1016        |
| <b>12 Zustandspruefung.....</b> | <b>1017</b> |
| GetLastError .....              | 1018        |
| IsStopped .....                 | 1019        |
| UninitializeReason .....        | 1020        |
| TerminalInfoInteger .....       | 1021        |
| TerminalInfoDouble .....        | 1022        |
| TerminalInfoString .....        | 1023        |
| MQLInfoInteger .....            | 1024        |
| MQLInfoString .....             | 1025        |
| Symbol .....                    | 1026        |
| Period .....                    | 1027        |
| Digits .....                    | 1028        |
| Point .....                     | 1029        |

|           |                                                                   |             |
|-----------|-------------------------------------------------------------------|-------------|
| <b>13</b> | <b>Marktinformation erhalten.....</b>                             | <b>1030</b> |
|           | SymbolsTotal .....                                                | 1032        |
|           | SymbolName .....                                                  | 1033        |
|           | SymbolSelect .....                                                | 1034        |
|           | SymbolsSynchronized .....                                         | 1035        |
|           | SymbolInfoDouble .....                                            | 1036        |
|           | SymbolInfoInteger .....                                           | 1038        |
|           | SymbolInfoString .....                                            | 1040        |
|           | SymbolInfoMarginRate .....                                        | 1042        |
|           | SymbolInfoTick .....                                              | 1043        |
|           | SymbolInfoSessionQuote .....                                      | 1044        |
|           | SymbolInfoSessionTrade .....                                      | 1045        |
|           | MarketBookAdd .....                                               | 1046        |
|           | MarketBookRelease .....                                           | 1047        |
|           | MarketBookGet .....                                               | 1048        |
| <b>14</b> | <b>Zugang zu Zeitreihen und Indikatoren.....</b>                  | <b>1049</b> |
|           | Richtung des Indizierens in Feldern, Puffern und Zeitreihen ..... | 1053        |
|           | Datenzugriff organisieren .....                                   | 1057        |
|           | SeriesInfoInteger .....                                           | 1067        |
|           | Bars .....                                                        | 1069        |
|           | BarsCalculated .....                                              | 1071        |
|           | IndicatorCreate .....                                             | 1073        |
|           | IndicatorParameters .....                                         | 1075        |
|           | IndicatorRelease .....                                            | 1077        |
|           | CopyBuffer .....                                                  | 1079        |
|           | CopyRates .....                                                   | 1084        |
|           | CopyTime .....                                                    | 1088        |
|           | CopyOpen .....                                                    | 1091        |
|           | CopyHigh .....                                                    | 1094        |
|           | CopyLow .....                                                     | 1098        |
|           | CopyClose .....                                                   | 1101        |
|           | CopyTickVolume .....                                              | 1104        |
|           | CopyRealVolume .....                                              | 1108        |
|           | CopySpread .....                                                  | 1111        |
|           | CopyTicks .....                                                   | 1115        |
| <b>15</b> | <b>Operationen mit Charts.....</b>                                | <b>1117</b> |
|           | ChartApplyTemplate .....                                          | 1120        |
|           | ChartSaveTemplate .....                                           | 1123        |
|           | ChartWindowFind .....                                             | 1128        |
|           | ChartTimePriceToXY .....                                          | 1130        |
|           | ChartXYToTimePrice .....                                          | 1131        |
|           | ChartOpen .....                                                   | 1133        |
|           | ChartFirst .....                                                  | 1134        |
|           | ChartNext .....                                                   | 1135        |
|           | ChartClose .....                                                  | 1136        |
|           | ChartSymbol .....                                                 | 1137        |
|           | ChartPeriod .....                                                 | 1138        |
|           | ChartRedraw .....                                                 | 1139        |
|           | ChartSetDouble .....                                              | 1140        |
|           | ChartSetInteger .....                                             | 1141        |
|           | ChartSetString .....                                              | 1142        |
|           | ChartGetDouble .....                                              | 1144        |
|           | ChartGetInteger .....                                             | 1146        |
|           | ChartGetString .....                                              | 1148        |
|           | ChartNavigate .....                                               | 1150        |
|           | ChartID .....                                                     | 1153        |
|           | ChartIndicatorAdd .....                                           | 1154        |

|                                                     |             |
|-----------------------------------------------------|-------------|
| ChartIndicatorDelete .....                          | 1157        |
| ChartIndicatorGet .....                             | 1160        |
| ChartIndicatorName .....                            | 1162        |
| ChartIndicatorsTotal .....                          | 1163        |
| ChartWindow OnDropped .....                         | 1164        |
| ChartPrice OnDropped .....                          | 1165        |
| ChartTime OnDropped .....                           | 1166        |
| ChartX OnDropped .....                              | 1167        |
| ChartY OnDropped .....                              | 1168        |
| ChartSetSymbolPeriod .....                          | 1169        |
| ChartScreenShot .....                               | 1170        |
| <b>16 Handelsfunktionen .....</b>                   | <b>1173</b> |
| OrderCalcMargin .....                               | 1175        |
| OrderCalcProfit .....                               | 1176        |
| OrderCheck .....                                    | 1177        |
| OrderSend .....                                     | 1178        |
| OrderSendAsync .....                                | 1183        |
| PositionsTotal .....                                | 1194        |
| PositionGetSymbol .....                             | 1195        |
| PositionSelect .....                                | 1196        |
| PositionGetDouble .....                             | 1197        |
| PositionGetInteger .....                            | 1198        |
| PositionGetString .....                             | 1200        |
| OrdersTotal .....                                   | 1201        |
| OrderGetTicket .....                                | 1202        |
| OrderSelect .....                                   | 1204        |
| OrderGetDouble .....                                | 1205        |
| OrderGetInteger .....                               | 1206        |
| OrderGetString .....                                | 1207        |
| HistorySelect .....                                 | 1208        |
| HistorySelectByPosition .....                       | 1210        |
| HistoryOrderSelect .....                            | 1211        |
| HistoryOrdersTotal .....                            | 1212        |
| HistoryOrderGetTicket .....                         | 1213        |
| HistoryOrderGetDouble .....                         | 1215        |
| HistoryOrderGetInteger .....                        | 1216        |
| HistoryOrderGetString .....                         | 1219        |
| HistoryDealSelect .....                             | 1220        |
| HistoryDealsTotal .....                             | 1221        |
| HistoryDealGetTicket .....                          | 1222        |
| HistoryDealGetDouble .....                          | 1225        |
| HistoryDealGetInteger .....                         | 1226        |
| HistoryDealGetString .....                          | 1229        |
| <b>17 Trade Signals .....</b>                       | <b>1230</b> |
| SignalBaseGetDouble .....                           | 1231        |
| SignalBaseGetInteger .....                          | 1232        |
| SignalBaseGetString .....                           | 1233        |
| SignalBaseSelect .....                              | 1234        |
| SignalBaseTotal .....                               | 1235        |
| SignalInfoGetDouble .....                           | 1236        |
| SignalInfoGetInteger .....                          | 1237        |
| SignalInfoGetString .....                           | 1238        |
| SignalInfoSetDouble .....                           | 1239        |
| SignalInfoSetInteger .....                          | 1240        |
| SignalSubscribe .....                               | 1241        |
| SignalUnsubscribe .....                             | 1242        |
| <b>18 Globalvariablen des Kundenterminals .....</b> | <b>1243</b> |

|                                             |             |
|---------------------------------------------|-------------|
| GlobalVariableCheck .....                   | 1244        |
| GlobalVariableTime .....                    | 1245        |
| GlobalVariableDel .....                     | 1246        |
| GlobalVariableGet .....                     | 1247        |
| GlobalVariableName .....                    | 1248        |
| GlobalVariableSet .....                     | 1249        |
| GlobalVariablesFlush .....                  | 1250        |
| GlobalVariableTemp .....                    | 1251        |
| GlobalVariableSetOnCondition .....          | 1252        |
| GlobalVariablesDeleteAll .....              | 1253        |
| GlobalVariablesTotal .....                  | 1254        |
| <b>19 Dateioperationen.....</b>             | <b>1255</b> |
| FileFindFirst .....                         | 1258        |
| FileFindNext .....                          | 1260        |
| FileFindClose .....                         | 1261        |
| FileIsExist .....                           | 1262        |
| FileOpen .....                              | 1265        |
| FileClose .....                             | 1268        |
| FileCopy .....                              | 1269        |
| FileDelete .....                            | 1272        |
| FileMove .....                              | 1275        |
| FileFlush .....                             | 1277        |
| FileGetInteger .....                        | 1279        |
| FileIsEnding .....                          | 1282        |
| FileIsLineEnding .....                      | 1283        |
| FileReadArray .....                         | 1289        |
| FileReadBool .....                          | 1291        |
| FileReadDatetime .....                      | 1295        |
| FileReadDouble .....                        | 1299        |
| FileReadFloat .....                         | 1302        |
| FileReadInteger .....                       | 1306        |
| FileReadLong .....                          | 1310        |
| FileReadNumber .....                        | 1313        |
| FileReadString .....                        | 1319        |
| FileReadStruct .....                        | 1321        |
| FileSeek .....                              | 1325        |
| FileSize .....                              | 1328        |
| FileTell .....                              | 1331        |
| FileWrite .....                             | 1334        |
| FileWriteArray .....                        | 1337        |
| FileWriteDouble .....                       | 1340        |
| FileWriteFloat .....                        | 1343        |
| FileWriteInteger .....                      | 1346        |
| FileWriteLong .....                         | 1349        |
| FileWriteString .....                       | 1352        |
| FileWriteStruct .....                       | 1355        |
| FolderCreate .....                          | 1358        |
| FolderDelete .....                          | 1361        |
| FolderClean .....                           | 1364        |
| <b>20 Benutzerindikatoren.....</b>          | <b>1367</b> |
| Stile der Indikator in den Beispielen ..... | 1371        |
| DRAW_NONE.....                              | 1379        |
| DRAW_LINE.....                              | 1382        |
| DRAW_SECTION.....                           | 1386        |
| DRAW_HISTOGRAM.....                         | 1390        |
| DRAW_HISTOGRAM2.....                        | 1394        |
| DRAW_ARROW .....                            | 1398        |
| DRAW_ZIGZAG.....                            | 1403        |

|                                                                                        |             |
|----------------------------------------------------------------------------------------|-------------|
| DRAW_FILLING .....                                                                     | 1408        |
| DRAW_BARS .....                                                                        | 1413        |
| DRAW_CANDLES .....                                                                     | 1419        |
| DRAW_COLOR_LINE .....                                                                  | 1425        |
| DRAW_COLOR_SECTION .....                                                               | 1430        |
| DRAW_COLOR_HISTOGRAM .....                                                             | 1436        |
| DRAW_COLOR_HISTOGRAM2 .....                                                            | 1441        |
| DRAW_COLOR_ARROW .....                                                                 | 1446        |
| DRAW_COLOR_ZIGZAG .....                                                                | 1452        |
| DRAW_COLOR_BARS .....                                                                  | 1457        |
| DRAW_COLOR_CANDLES .....                                                               | 1464        |
| Zusammenhang zwischen Eigenschaften des Indikators und entsprechenden Funktionen ..... | 1471        |
| SetIndexBuffer .....                                                                   | 1474        |
| IndicatorSetDouble .....                                                               | 1477        |
| IndicatorSetInteger .....                                                              | 1481        |
| IndicatorSetString .....                                                               | 1485        |
| PlotIndexSetDouble .....                                                               | 1488        |
| PlotIndexSetInteger .....                                                              | 1489        |
| PlotIndexSetString .....                                                               | 1493        |
| PlotIndexGetInteger .....                                                              | 1494        |
| <b>21 Graphische Objekte .....</b>                                                     | <b>1497</b> |
| ObjectCreate .....                                                                     | 1499        |
| ObjectName .....                                                                       | 1503        |
| ObjectDelete .....                                                                     | 1504        |
| ObjectsDeleteAll .....                                                                 | 1505        |
| ObjectFind .....                                                                       | 1506        |
| ObjectGetTimeByValue .....                                                             | 1507        |
| ObjectGetValueByTime .....                                                             | 1508        |
| ObjectMove .....                                                                       | 1509        |
| ObjectsTotal .....                                                                     | 1510        |
| ObjectSetDouble .....                                                                  | 1511        |
| ObjectSetInteger .....                                                                 | 1514        |
| ObjectSetString .....                                                                  | 1517        |
| ObjectGetDouble .....                                                                  | 1519        |
| ObjectGetInteger .....                                                                 | 1520        |
| ObjectGetString .....                                                                  | 1521        |
| TextSetFont .....                                                                      | 1523        |
| TextOut .....                                                                          | 1526        |
| TextGetSize .....                                                                      | 1530        |
| <b>22 Technische Indikatoren .....</b>                                                 | <b>1531</b> |
| iAC .....                                                                              | 1533        |
| iAD .....                                                                              | 1538        |
| iADX .....                                                                             | 1543        |
| iADXWilder .....                                                                       | 1548        |
| iAlligator .....                                                                       | 1553        |
| iAMA .....                                                                             | 1560        |
| iAO .....                                                                              | 1565        |
| iATR .....                                                                             | 1570        |
| iBearsPower .....                                                                      | 1575        |
| iBands .....                                                                           | 1580        |
| iBullsPower .....                                                                      | 1586        |
| iCCI .....                                                                             | 1591        |
| iChaikin .....                                                                         | 1596        |
| iCustom .....                                                                          | 1601        |
| iDEMA .....                                                                            | 1604        |
| iDeMarker .....                                                                        | 1609        |
| iEnvelopes .....                                                                       | 1614        |
| iForce .....                                                                           | 1620        |

|                                                      |             |
|------------------------------------------------------|-------------|
| iFractals .....                                      | 1625        |
| iFrAMA .....                                         | 1630        |
| iGator .....                                         | 1635        |
| ilchimoku .....                                      | 1642        |
| iBWMFI .....                                         | 1649        |
| iMomentum .....                                      | 1654        |
| iMFI .....                                           | 1659        |
| iMA .....                                            | 1664        |
| iOsMA .....                                          | 1669        |
| iMACD .....                                          | 1674        |
| iOBV .....                                           | 1680        |
| iSAR .....                                           | 1685        |
| iRSI .....                                           | 1690        |
| iRVI .....                                           | 1695        |
| iStdDev .....                                        | 1700        |
| iStochastic .....                                    | 1705        |
| iTEMA .....                                          | 1711        |
| iTriX .....                                          | 1716        |
| iWPR .....                                           | 1721        |
| iVIDyA .....                                         | 1726        |
| iVolumes .....                                       | 1731        |
| <b>23 Arbeit mit Ergebnisse der Optimierung.....</b> | <b>1736</b> |
| FrameFirst .....                                     | 1737        |
| FrameFilter .....                                    | 1738        |
| FrameNext .....                                      | 1739        |
| FrameInputs .....                                    | 1740        |
| FrameAdd .....                                       | 1741        |
| ParameterGetRange .....                              | 1742        |
| ParameterSetRange .....                              | 1746        |
| <b>24 Arbeit mit Ereignissen .....</b>               | <b>1748</b> |
| EventSetMillisecondTimer .....                       | 1749        |
| EventSetTimer .....                                  | 1750        |
| EventKillTimer .....                                 | 1751        |
| EventChartCustom .....                               | 1752        |
| <b>25 Arbeit mit OpenCL.....</b>                     | <b>1758</b> |
| CLHandleType .....                                   | 1759        |
| CLGetInfoInteger .....                               | 1760        |
| CLGetInfoString .....                                | 1762        |
| CLContextCreate .....                                | 1765        |
| CLContextFree .....                                  | 1766        |
| CLGetDeviceInfo .....                                | 1767        |
| CLProgramCreate .....                                | 1771        |
| CLProgramFree .....                                  | 1772        |
| CLKernelCreate .....                                 | 1773        |
| CLKernelFree .....                                   | 1774        |
| CLSetKernelArg .....                                 | 1775        |
| CLSetKernelArgMem .....                              | 1776        |
| CLBufferCreate .....                                 | 1777        |
| CLBufferFree .....                                   | 1778        |
| CLBufferWrite .....                                  | 1779        |
| CLBufferRead .....                                   | 1780        |
| CLExecute .....                                      | 1781        |
| <b>26 Standardbibliothek.....</b>                    | <b>1783</b> |
| Basic Class CObject .....                            | 1784        |
| Prev .....                                           | 1785        |
| Prev .....                                           | 1786        |
| Next .....                                           | 1787        |



|                              |             |
|------------------------------|-------------|
| Next .....                   | 1788        |
| Compare.....                 | 1789        |
| Save .....                   | 1791        |
| Load .....                   | 1793        |
| Type .....                   | 1795        |
| <b>Classes of data .....</b> | <b>1796</b> |
| CArray .....                 | 1797        |
| Step .....                   | 1799        |
| Step .....                   | 1800        |
| Total .....                  | 1801        |
| Available .....              | 1802        |
| Max .....                    | 1803        |
| IsSorted .....               | 1804        |
| SortMode .....               | 1805        |
| Clear .....                  | 1806        |
| Sort .....                   | 1807        |
| Save .....                   | 1808        |
| Load .....                   | 1809        |
| CArrayChar .....             | 1810        |
| Reserve .....                | 1812        |
| Resize .....                 | 1813        |
| Shutdown .....               | 1814        |
| Add .....                    | 1815        |
| AddArray .....               | 1816        |
| AddArray .....               | 1817        |
| Insert .....                 | 1819        |
| InsertArray.....             | 1820        |
| InsertArray.....             | 1821        |
| AssignArray.....             | 1823        |
| AssignArray.....             | 1824        |
| Update .....                 | 1826        |
| Shift .....                  | 1827        |
| Delete .....                 | 1828        |
| DeleteRange.....             | 1829        |
| At .....                     | 1830        |
| CompareArray .....           | 1832        |
| CompareArray .....           | 1833        |
| InsertSort .....             | 1834        |
| Search .....                 | 1835        |
| SearchGreat.....             | 1836        |
| SearchLess .....             | 1837        |
| SearchGreatOrEqual.....      | 1838        |
| SearchLessOrEqual.....       | 1839        |
| SearchFirst.....             | 1840        |
| SearchLast .....             | 1841        |
| SearchLinear.....            | 1842        |
| Save .....                   | 1843        |
| Load .....                   | 1844        |
| Type .....                   | 1846        |
| CArrayShort.....             | 1847        |
| Reserve .....                | 1849        |
| Resize .....                 | 1850        |
| Shutdown .....               | 1851        |
| Add .....                    | 1852        |
| AddArray .....               | 1853        |
| AddArray .....               | 1854        |
| Insert .....                 | 1856        |
| InsertArray.....             | 1857        |

|                         |      |
|-------------------------|------|
| InsertArray.....        | 1858 |
| AssignArray.....        | 1860 |
| AssignArray.....        | 1861 |
| Update .....            | 1863 |
| Shift .....             | 1864 |
| Delete .....            | 1865 |
| DeleteRange.....        | 1866 |
| At .....                | 1867 |
| CompareArray .....      | 1869 |
| CompareArray .....      | 1870 |
| InsertSort .....        | 1871 |
| Search .....            | 1872 |
| SearchGreat.....        | 1873 |
| SearchLess .....        | 1874 |
| SearchGreatOrEqual..... | 1875 |
| SearchLessOrEqual.....  | 1876 |
| SearchFirst.....        | 1877 |
| SearchLast .....        | 1878 |
| SearchLinear.....       | 1879 |
| Save .....              | 1880 |
| Load .....              | 1882 |
| Type .....              | 1884 |
| CArrayInt.....          | 1885 |
| Reserve .....           | 1887 |
| Resize .....            | 1888 |
| Shutdown .....          | 1889 |
| Add .....               | 1890 |
| AddArray .....          | 1891 |
| AddArray .....          | 1892 |
| Insert .....            | 1894 |
| InsertArray.....        | 1895 |
| InsertArray.....        | 1896 |
| AssignArray.....        | 1898 |
| AssignArray.....        | 1899 |
| Update .....            | 1901 |
| Shift .....             | 1902 |
| Delete .....            | 1903 |
| DeleteRange.....        | 1904 |
| At .....                | 1905 |
| CompareArray .....      | 1907 |
| CompareArray .....      | 1908 |
| InsertSort .....        | 1909 |
| Search .....            | 1910 |
| SearchGreat.....        | 1911 |
| SearchLess .....        | 1912 |
| SearchGreatOrEqual..... | 1913 |
| SearchLessOrEqual.....  | 1914 |
| SearchFirst.....        | 1915 |
| SearchLast .....        | 1916 |
| SearchLinear.....       | 1917 |
| Save .....              | 1918 |
| Load .....              | 1920 |
| Type .....              | 1922 |
| CArrayLong .....        | 1923 |
| Reserve .....           | 1925 |
| Resize .....            | 1926 |
| Shutdown .....          | 1927 |
| Add .....               | 1928 |

|                         |      |
|-------------------------|------|
| AddArray .....          | 1929 |
| AddArray .....          | 1930 |
| Insert .....            | 1932 |
| InsertArray.....        | 1933 |
| InsertArray.....        | 1934 |
| AssignArray.....        | 1936 |
| AssignArray.....        | 1937 |
| Update .....            | 1939 |
| Shift .....             | 1940 |
| Delete .....            | 1941 |
| DeleteRange.....        | 1942 |
| At .....                | 1943 |
| CompareArray .....      | 1945 |
| CompareArray .....      | 1946 |
| InsertSort .....        | 1947 |
| Search .....            | 1948 |
| SearchGreat.....        | 1949 |
| SearchLess .....        | 1950 |
| SearchGreatOrEqual..... | 1951 |
| SearchLessOrEqual.....  | 1952 |
| SearchFirst.....        | 1953 |
| SearchLast .....        | 1954 |
| SearchLinear.....       | 1955 |
| Save .....              | 1956 |
| Load .....              | 1958 |
| Type .....              | 1960 |
| CArrayFloat.....        | 1961 |
| Delta .....             | 1963 |
| Reserve .....           | 1964 |
| Resize .....            | 1965 |
| Shutdown .....          | 1966 |
| Add .....               | 1967 |
| AddArray .....          | 1968 |
| AddArray .....          | 1969 |
| Insert .....            | 1971 |
| InsertArray.....        | 1972 |
| InsertArray.....        | 1973 |
| AssignArray.....        | 1975 |
| AssignArray.....        | 1976 |
| Update .....            | 1978 |
| Shift .....             | 1979 |
| Delete .....            | 1980 |
| DeleteRange.....        | 1981 |
| At .....                | 1982 |
| CompareArray .....      | 1984 |
| CompareArray .....      | 1985 |
| InsertSort .....        | 1986 |
| Search .....            | 1987 |
| SearchGreat.....        | 1988 |
| SearchLess .....        | 1989 |
| SearchGreatOrEqual..... | 1990 |
| SearchLessOrEqual.....  | 1991 |
| SearchFirst.....        | 1992 |
| SearchLast .....        | 1993 |
| SearchLinear.....       | 1994 |
| Save .....              | 1995 |
| Load .....              | 1997 |
| Type .....              | 1999 |

|                         |      |
|-------------------------|------|
| CArrayDouble.....       | 2000 |
| Delta .....             | 2003 |
| Reserve .....           | 2004 |
| Resize .....            | 2005 |
| Shutdown .....          | 2006 |
| Add .....               | 2007 |
| AddArray .....          | 2008 |
| AddArray .....          | 2009 |
| Insert .....            | 2011 |
| InsertArray.....        | 2012 |
| InsertArray.....        | 2013 |
| AssignArray.....        | 2015 |
| AssignArray.....        | 2016 |
| Update .....            | 2018 |
| Shift .....             | 2019 |
| Delete .....            | 2020 |
| DeleteRange.....        | 2021 |
| At .....                | 2022 |
| CompareArray .....      | 2024 |
| CompareArray .....      | 2025 |
| Minimum .....           | 2026 |
| Maximum .....           | 2027 |
| InsertSort .....        | 2028 |
| Search .....            | 2029 |
| SearchGreat.....        | 2030 |
| SearchLess .....        | 2031 |
| SearchGreatOrEqual..... | 2032 |
| SearchLessOrEqual.....  | 2033 |
| SearchFirst.....        | 2034 |
| SearchLast .....        | 2035 |
| SearchLinear.....       | 2036 |
| Save .....              | 2037 |
| Load .....              | 2039 |
| Type .....              | 2041 |
| CArrayString.....       | 2042 |
| Reserve .....           | 2044 |
| Resize .....            | 2045 |
| Shutdown .....          | 2046 |
| Add .....               | 2047 |
| AddArray .....          | 2048 |
| AddArray .....          | 2049 |
| Insert .....            | 2051 |
| InsertArray.....        | 2052 |
| InsertArray.....        | 2053 |
| AssignArray.....        | 2055 |
| AssignArray.....        | 2056 |
| Update .....            | 2058 |
| Shift .....             | 2059 |
| Delete .....            | 2060 |
| DeleteRange.....        | 2061 |
| At .....                | 2062 |
| CompareArray .....      | 2064 |
| CompareArray .....      | 2065 |
| InsertSort .....        | 2066 |
| Search .....            | 2067 |
| SearchGreat.....        | 2068 |
| SearchLess .....        | 2069 |
| SearchGreatOrEqual..... | 2070 |

|                         |      |
|-------------------------|------|
| SearchLessOrEqual.....  | 2071 |
| SearchFirst.....        | 2072 |
| SearchLast.....         | 2073 |
| SearchLinear.....       | 2074 |
| Save.....               | 2075 |
| Load.....               | 2077 |
| Type.....               | 2079 |
| CArrayObj.....          | 2080 |
| FreeMode.....           | 2085 |
| FreeMode.....           | 2086 |
| Reserve.....            | 2088 |
| Resize.....             | 2089 |
| Clear.....              | 2091 |
| Shutdown.....           | 2092 |
| CreateElement.....      | 2093 |
| Add.....                | 2095 |
| AddArray.....           | 2096 |
| Insert.....             | 2099 |
| InsertArray.....        | 2101 |
| AssignArray.....        | 2103 |
| Update.....             | 2105 |
| Shift.....              | 2106 |
| Detach.....             | 2107 |
| Delete.....             | 2108 |
| DeleteRange.....        | 2109 |
| At.....                 | 2110 |
| CompareArray.....       | 2111 |
| InsertSort.....         | 2112 |
| Search.....             | 2113 |
| SearchGreat.....        | 2114 |
| SearchLess.....         | 2115 |
| SearchGreatOrEqual..... | 2116 |
| SearchLessOrEqual.....  | 2117 |
| SearchFirst.....        | 2118 |
| SearchLast.....         | 2119 |
| Save.....               | 2120 |
| Load.....               | 2121 |
| Type.....               | 2123 |
| CList.....              | 2124 |
| FreeMode.....           | 2126 |
| FreeMode.....           | 2127 |
| Total.....              | 2129 |
| IsSorted.....           | 2130 |
| SortMode.....           | 2131 |
| CreateElement.....      | 2132 |
| Add.....                | 2133 |
| Insert.....             | 2134 |
| DetachCurrent.....      | 2136 |
| DeleteCurrent.....      | 2137 |
| Delete.....             | 2138 |
| Clear.....              | 2139 |
| IndexOf.....            | 2140 |
| GetNodeAtIndex.....     | 2141 |
| GetFirstNode.....       | 2142 |
| GetPrevNode.....        | 2143 |
| GetCurrentNode.....     | 2144 |
| GetNextNode.....        | 2145 |
| GetLastNode.....        | 2146 |

|                                          |             |
|------------------------------------------|-------------|
| Sort .....                               | 2147        |
| MoveToIndex.....                         | 2148        |
| Exchange .....                           | 2149        |
| CompareList.....                         | 2150        |
| Search .....                             | 2151        |
| Save .....                               | 2152        |
| Load .....                               | 2154        |
| Type .....                               | 2156        |
| CTreeNode .....                          | 2157        |
| Owner .....                              | 2162        |
| Left .....                               | 2163        |
| Right .....                              | 2164        |
| Balance .....                            | 2165        |
| BalanceL .....                           | 2166        |
| BalanceR .....                           | 2167        |
| CreateSample .....                       | 2168        |
| RefreshBalance .....                     | 2169        |
| GetNext .....                            | 2170        |
| SaveNode .....                           | 2171        |
| LoadNode .....                           | 2172        |
| Type .....                               | 2173        |
| CTree .....                              | 2174        |
| Root .....                               | 2179        |
| CreateElement.....                       | 2180        |
| Insert .....                             | 2181        |
| Detach .....                             | 2182        |
| Delete .....                             | 2183        |
| Clear .....                              | 2184        |
| Find .....                               | 2185        |
| Save .....                               | 2186        |
| Load .....                               | 2187        |
| Type .....                               | 2188        |
| <b>Classes for Graphic Objects .....</b> | <b>2189</b> |
| CChartObject .....                       | 2190        |
| ChartId .....                            | 2193        |
| Window .....                             | 2194        |
| Name .....                               | 2195        |
| NumPoints .....                          | 2196        |
| Attach .....                             | 2197        |
| SetPoint .....                           | 2198        |
| Delete .....                             | 2199        |
| Detach .....                             | 2200        |
| ShiftObject.....                         | 2201        |
| ShiftPoint .....                         | 2202        |
| Time .....                               | 2203        |
| Price .....                              | 2205        |
| Color .....                              | 2207        |
| Style .....                              | 2208        |
| Width .....                              | 2209        |
| Background.....                          | 2210        |
| Selected .....                           | 2211        |
| Selectable .....                         | 2212        |
| Description.....                         | 2213        |
| Tooltip .....                            | 2214        |
| Timeframes.....                          | 2215        |
| Z_Order .....                            | 2216        |
| CreateTime.....                          | 2217        |
| LevelsCount .....                        | 2218        |

|                                |      |
|--------------------------------|------|
| LevelColor .....               | 2219 |
| LevelStyle .....               | 2221 |
| LevelWidth .....               | 2223 |
| LevelValue .....               | 2225 |
| LevelDescription .....         | 2227 |
| GetInteger .....               | 2229 |
| SetInteger .....               | 2231 |
| GetDouble .....                | 2233 |
| SetDouble .....                | 2235 |
| GetString .....                | 2237 |
| SetString .....                | 2239 |
| Save .....                     | 2241 |
| Load .....                     | 2242 |
| Type .....                     | 2243 |
| Objects Lines.....             | 2244 |
| CChartObjectVLine.....         | 2245 |
| Create .....                   | 2246 |
| Type .....                     | 2247 |
| CChartObjectHLine.....         | 2248 |
| Create .....                   | 2249 |
| Type .....                     | 2250 |
| CChartObjectTrend .....        | 2251 |
| Create .....                   | 2252 |
| RayLeft .....                  | 2253 |
| RayRight .....                 | 2254 |
| Save .....                     | 2255 |
| Load .....                     | 2256 |
| Type .....                     | 2257 |
| CChartObjectTrendByAngle.....  | 2258 |
| Create .....                   | 2259 |
| Angle .....                    | 2260 |
| Type .....                     | 2261 |
| CChartObjectCycles.....        | 2262 |
| Create .....                   | 2263 |
| Type .....                     | 2264 |
| Objects Channels .....         | 2265 |
| CChartObjectChannel.....       | 2266 |
| Create .....                   | 2267 |
| Type .....                     | 2268 |
| CChartObjectRegression.....    | 2269 |
| Create .....                   | 2270 |
| Type .....                     | 2271 |
| CChartObjectStdDevChannel..... | 2272 |
| Create .....                   | 2273 |
| Deviations.....                | 2274 |
| Save .....                     | 2275 |
| Load .....                     | 2276 |
| Type .....                     | 2277 |
| CChartObjectPitchfork.....     | 2278 |
| Create .....                   | 2279 |
| Type .....                     | 2280 |
| Gann Tools.....                | 2281 |
| CChartObjectGannLine.....      | 2282 |
| Create .....                   | 2283 |
| PipsPerBar.....                | 2284 |
| Save .....                     | 2285 |
| Load .....                     | 2286 |
| Type .....                     | 2287 |

|                                |      |
|--------------------------------|------|
| CChartObjectGannFan.....       | 2288 |
| Create .....                   | 2289 |
| PipsPerBar.....                | 2290 |
| Downtrend.....                 | 2291 |
| Save .....                     | 2292 |
| Load .....                     | 2293 |
| Type .....                     | 2294 |
| CChartObjectGannGrid.....      | 2295 |
| Create .....                   | 2296 |
| PipsPerBar.....                | 2297 |
| Downtrend.....                 | 2298 |
| Save .....                     | 2299 |
| Load .....                     | 2300 |
| Type .....                     | 2301 |
| Fibonacci Tools.....           | 2302 |
| CChartObjectFibo .....         | 2303 |
| Create .....                   | 2304 |
| Type .....                     | 2305 |
| CChartObjectFiboTimes .....    | 2306 |
| Create .....                   | 2307 |
| Type .....                     | 2308 |
| CChartObjectFiboFan.....       | 2309 |
| Create .....                   | 2310 |
| Type .....                     | 2311 |
| CChartObjectFiboArc.....       | 2312 |
| Create .....                   | 2313 |
| Scale .....                    | 2314 |
| Ellipse .....                  | 2315 |
| Save .....                     | 2316 |
| Load .....                     | 2317 |
| Type .....                     | 2318 |
| CChartObjectFiboChannel.....   | 2319 |
| Create .....                   | 2320 |
| Type .....                     | 2321 |
| CChartObjectFiboExpansion..... | 2322 |
| Create .....                   | 2323 |
| Type .....                     | 2324 |
| Elliott Tools.....             | 2325 |
| CChartObjectElliottWave3.....  | 2326 |
| Create .....                   | 2327 |
| Degree .....                   | 2328 |
| Lines .....                    | 2329 |
| Save .....                     | 2330 |
| Load .....                     | 2331 |
| Type .....                     | 2332 |
| CChartObjectElliottWave5.....  | 2333 |
| Create .....                   | 2334 |
| Type .....                     | 2336 |
| Objects Shapes.....            | 2337 |
| CChartObjectRectangle.....     | 2338 |
| Create .....                   | 2339 |
| Type .....                     | 2340 |
| CChartObjectTriangle.....      | 2341 |
| Create .....                   | 2342 |
| Type .....                     | 2343 |
| CChartObjectEllipse.....       | 2344 |
| Create .....                   | 2345 |
| Type .....                     | 2346 |



|                             |      |
|-----------------------------|------|
| Objects Arrows.....         | 2347 |
| CChartObjectArrow.....      | 2348 |
| Create .....                | 2349 |
| ArrowCode.....              | 2351 |
| Anchor .....                | 2353 |
| Save .....                  | 2355 |
| Load .....                  | 2356 |
| Type .....                  | 2357 |
| Arrows with fixed code..... | 2358 |
| Create .....                | 2360 |
| ArrowCode.....              | 2362 |
| Type .....                  | 2363 |
| Objects Controls.....       | 2364 |
| CChartObjectText.....       | 2365 |
| Create .....                | 2366 |
| Angle .....                 | 2367 |
| Font .....                  | 2368 |
| FontSize .....              | 2369 |
| Anchor .....                | 2370 |
| Save .....                  | 2371 |
| Load .....                  | 2372 |
| Type .....                  | 2373 |
| CChartObjectLabel.....      | 2374 |
| Create .....                | 2375 |
| X_Distance.....             | 2376 |
| Y_Distance.....             | 2377 |
| X_Size .....                | 2378 |
| Y_Size .....                | 2379 |
| Corner .....                | 2380 |
| Time .....                  | 2381 |
| Price .....                 | 2382 |
| Save .....                  | 2383 |
| Load .....                  | 2384 |
| Type .....                  | 2385 |
| CChartObjectEdit.....       | 2386 |
| Create .....                | 2387 |
| TextAlign .....             | 2388 |
| X_Size .....                | 2389 |
| Y_Size .....                | 2390 |
| BackColor .....             | 2391 |
| BorderColor .....           | 2392 |
| ReadOnly .....              | 2393 |
| Angle .....                 | 2394 |
| Save .....                  | 2395 |
| Load .....                  | 2396 |
| Type .....                  | 2397 |
| CChartObjectButton.....     | 2398 |
| State .....                 | 2399 |
| Save .....                  | 2400 |
| Load .....                  | 2401 |
| Type .....                  | 2402 |
| CChartObjectSubChart.....   | 2403 |
| Create .....                | 2405 |
| X_Distance.....             | 2406 |
| Y_Distance.....             | 2407 |
| Corner .....                | 2408 |
| X_Size .....                | 2409 |
| Y_Size .....                | 2410 |

|                                                 |             |
|-------------------------------------------------|-------------|
| Symbol .....                                    | 2411        |
| Period .....                                    | 2412        |
| Scale .....                                     | 2413        |
| DateScale .....                                 | 2414        |
| PriceScale .....                                | 2415        |
| Time .....                                      | 2416        |
| Price .....                                     | 2417        |
| Save .....                                      | 2418        |
| Load .....                                      | 2419        |
| Type .....                                      | 2420        |
| CChartObjectBitmap .....                        | 2421        |
| Create .....                                    | 2422        |
| BmpFile .....                                   | 2423        |
| X_Offset .....                                  | 2424        |
| Y_Offset .....                                  | 2425        |
| Save .....                                      | 2426        |
| Load .....                                      | 2427        |
| Type .....                                      | 2428        |
| CChartObjectBmpLabel .....                      | 2429        |
| Create .....                                    | 2431        |
| X_Distance .....                                | 2432        |
| Y_Distance .....                                | 2433        |
| X_Offset .....                                  | 2434        |
| Y_Offset .....                                  | 2435        |
| Corner .....                                    | 2436        |
| X_Size .....                                    | 2437        |
| Y_Size .....                                    | 2438        |
| BmpFileOn .....                                 | 2439        |
| BmpFileOff .....                                | 2440        |
| State .....                                     | 2441        |
| Time .....                                      | 2442        |
| Price .....                                     | 2443        |
| Save .....                                      | 2444        |
| Load .....                                      | 2445        |
| Type .....                                      | 2446        |
| CChartObjectRectLabel .....                     | 2447        |
| Create .....                                    | 2448        |
| X_Size .....                                    | 2449        |
| Y_Size .....                                    | 2450        |
| BackColor .....                                 | 2451        |
| Angle .....                                     | 2452        |
| BorderType .....                                | 2453        |
| Save .....                                      | 2454        |
| Load .....                                      | 2455        |
| Type .....                                      | 2456        |
| <b>Class for creating custom graphics .....</b> | <b>2457</b> |
| ChartObjectName .....                           | 2460        |
| Circle .....                                    | 2461        |
| CircleAA .....                                  | 2462        |
| Create .....                                    | 2463        |
| CreateBitmap .....                              | 2464        |
| CreateBitmapLabel .....                         | 2466        |
| Destroy .....                                   | 2468        |
| Erase .....                                     | 2469        |
| Fill .....                                      | 2470        |
| FillCircle .....                                | 2471        |
| FillRectangle .....                             | 2472        |
| FillTriangle .....                              | 2473        |

|                                           |             |
|-------------------------------------------|-------------|
| FontAngleGet .....                        | 2474        |
| FontAngleSet .....                        | 2475        |
| FontFlagsGet .....                        | 2476        |
| FontFlagsSet .....                        | 2477        |
| FontGet .....                             | 2478        |
| FontNameGet .....                         | 2479        |
| FontNameSet .....                         | 2480        |
| FontSet .....                             | 2481        |
| FontSizeGet .....                         | 2482        |
| FontSizeSet .....                         | 2483        |
| Height .....                              | 2484        |
| Line .....                                | 2485        |
| LineAA .....                              | 2486        |
| LineHorizontal .....                      | 2487        |
| LineStyleSet .....                        | 2488        |
| LineVertical .....                        | 2489        |
| LoadFromFile .....                        | 2490        |
| PixelGet .....                            | 2491        |
| PixelSet .....                            | 2492        |
| PixelSetAA .....                          | 2493        |
| Polygon .....                             | 2494        |
| PolygonAA .....                           | 2495        |
| Polyline .....                            | 2496        |
| PolylineAA .....                          | 2497        |
| Rectangle .....                           | 2498        |
| Resize .....                              | 2499        |
| ResourceName .....                        | 2500        |
| TextHeight .....                          | 2501        |
| TextOut .....                             | 2502        |
| TextSize .....                            | 2503        |
| TextWidth .....                           | 2504        |
| TransparentLevelSet .....                 | 2505        |
| Triangle .....                            | 2506        |
| TriangleAA .....                          | 2507        |
| Update .....                              | 2508        |
| Width .....                               | 2509        |
| <b>Class for working with chart .....</b> | <b>2510</b> |
| ChartID .....                             | 2515        |
| Mode .....                                | 2516        |
| Foreground .....                          | 2517        |
| Shift .....                               | 2518        |
| ShiftSize .....                           | 2519        |
| AutoScroll .....                          | 2520        |
| Scale .....                               | 2521        |
| ScaleFix .....                            | 2522        |
| ScaleFix_11 .....                         | 2523        |
| FixedMax .....                            | 2524        |
| FixedMin .....                            | 2525        |
| PointsPerBar .....                        | 2526        |
| ScalePPB .....                            | 2527        |
| ShowOHLC .....                            | 2528        |
| ShowLineBid .....                         | 2529        |
| ShowLineAsk .....                         | 2530        |
| ShowLastLine .....                        | 2531        |
| ShowPeriodSep .....                       | 2532        |
| ShowGrid .....                            | 2533        |
| ShowVolumes .....                         | 2534        |
| ShowObjectDescr .....                     | 2535        |

|                        |      |
|------------------------|------|
| ShowDateScale.....     | 2536 |
| ShowPriceScale.....    | 2537 |
| ColorBackground.....   | 2538 |
| ColorForeground.....   | 2539 |
| ColorGrid.....         | 2540 |
| ColorBarUp.....        | 2541 |
| ColorBarDown.....      | 2542 |
| ColorCandleBull.....   | 2543 |
| ColorCandleBear.....   | 2544 |
| ColorChartLine.....    | 2545 |
| ColorVolumes.....      | 2546 |
| ColorLineBid.....      | 2547 |
| ColorLineAsk.....      | 2548 |
| ColorLineLast.....     | 2549 |
| ColorStopLevels.....   | 2550 |
| VisibleBars.....       | 2551 |
| WindowsTotal.....      | 2552 |
| WindowsVisible.....    | 2553 |
| WindowHandle.....      | 2554 |
| FirstVisibleBar.....   | 2555 |
| WidthInBars.....       | 2556 |
| WidthInPixels.....     | 2557 |
| HeightInPixels.....    | 2558 |
| PriceMin.....          | 2559 |
| PriceMax.....          | 2560 |
| Attach.....            | 2561 |
| FirstChart.....        | 2562 |
| NextChart.....         | 2563 |
| Open.....              | 2564 |
| Detach.....            | 2565 |
| Close.....             | 2566 |
| BringToTop.....        | 2567 |
| EventObjectCreate..... | 2568 |
| EventObjectDelete..... | 2569 |
| IndicatorAdd.....      | 2570 |
| IndicatorDelete.....   | 2571 |
| IndicatorsTotal.....   | 2572 |
| IndicatorName.....     | 2573 |
| Navigate.....          | 2574 |
| Symbol.....            | 2575 |
| Period.....            | 2576 |
| Redraw.....            | 2577 |
| GetInteger.....        | 2578 |
| SetInteger.....        | 2579 |
| GetDouble.....         | 2580 |
| SetDouble.....         | 2581 |
| GetString.....         | 2582 |
| SetString.....         | 2583 |
| SetSymbolPeriod.....   | 2584 |
| ApplyTemplate.....     | 2585 |
| ScreenShot.....        | 2586 |
| WindowOnDropped.....   | 2587 |
| PriceOnDropped.....    | 2588 |
| TimeOnDropped.....     | 2589 |
| XOnDropped.....        | 2590 |
| YOnDropped.....        | 2591 |
| Save.....              | 2592 |
| Load.....              | 2593 |

|                                          |             |
|------------------------------------------|-------------|
| Type .....                               | 2594        |
| <b>Classes for file operations .....</b> | <b>2595</b> |
| CFile .....                              | 2596        |
| Handle .....                             | 2598        |
| Filename .....                           | 2599        |
| Flags .....                              | 2600        |
| SetUnicode .....                         | 2601        |
| SetCommon .....                          | 2602        |
| Open .....                               | 2603        |
| Close .....                              | 2604        |
| Delete .....                             | 2605        |
| IsExist .....                            | 2606        |
| Copy .....                               | 2607        |
| Move .....                               | 2608        |
| Size .....                               | 2609        |
| Tell .....                               | 2610        |
| Seek .....                               | 2611        |
| Flush .....                              | 2612        |
| IsEnding .....                           | 2613        |
| IsLineEnding .....                       | 2614        |
| FolderCreate .....                       | 2615        |
| FolderDelete .....                       | 2616        |
| FolderClean .....                        | 2617        |
| FileFindFirst .....                      | 2618        |
| FileFindNext .....                       | 2619        |
| FileFindClose .....                      | 2620        |
| CFileBin .....                           | 2621        |
| Open .....                               | 2623        |
| WriteChar .....                          | 2624        |
| WriteShort .....                         | 2625        |
| WriteInteger .....                       | 2626        |
| WriteLong .....                          | 2627        |
| WriteFloat .....                         | 2628        |
| WriteDouble .....                        | 2629        |
| WriteString .....                        | 2630        |
| WriteCharArray .....                     | 2631        |
| WriteShortArray .....                    | 2632        |
| WriteIntegerArray .....                  | 2633        |
| WriteLongArray .....                     | 2634        |
| WriteFloatArray .....                    | 2635        |
| WriteDoubleArray .....                   | 2636        |
| WriteObject .....                        | 2637        |
| ReadChar .....                           | 2638        |
| ReadShort .....                          | 2639        |
| ReadInteger .....                        | 2640        |
| ReadLong .....                           | 2641        |
| ReadFloat .....                          | 2642        |
| ReadDouble .....                         | 2643        |
| ReadString .....                         | 2644        |
| ReadCharArray .....                      | 2645        |
| ReadShortArray .....                     | 2646        |
| ReadIntegerArray .....                   | 2647        |
| ReadLongArray .....                      | 2648        |
| ReadFloatArray .....                     | 2649        |
| ReadDoubleArray .....                    | 2650        |
| ReadObject .....                         | 2651        |
| CFileTxt .....                           | 2652        |
| Open .....                               | 2653        |

|                                                  |             |
|--------------------------------------------------|-------------|
| WriteString.....                                 | 2654        |
| ReadString.....                                  | 2655        |
| <b>Class for String operations .....</b>         | <b>2656</b> |
| CString.....                                     | 2657        |
| Str .....                                        | 2659        |
| Len .....                                        | 2660        |
| Copy .....                                       | 2661        |
| Fill .....                                       | 2662        |
| Assign .....                                     | 2663        |
| Append .....                                     | 2664        |
| Insert .....                                     | 2665        |
| Compare .....                                    | 2666        |
| CompareNoCase.....                               | 2667        |
| Left .....                                       | 2668        |
| Right .....                                      | 2669        |
| Mid .....                                        | 2670        |
| Trim .....                                       | 2671        |
| TrimLeft .....                                   | 2672        |
| TrimRight .....                                  | 2673        |
| Clear .....                                      | 2674        |
| ToUpper .....                                    | 2675        |
| ToLower .....                                    | 2676        |
| Reverse .....                                    | 2677        |
| Find .....                                       | 2678        |
| FindRev .....                                    | 2679        |
| Remove .....                                     | 2680        |
| Replace .....                                    | 2681        |
| <b>Classes for working with Indicators .....</b> | <b>2682</b> |
| Base classes.....                                | 2683        |
| CSpreadBuffer .....                              | 2684        |
| Size .....                                       | 2685        |
| SetSymbolPeriod.....                             | 2686        |
| At .....                                         | 2687        |
| Refresh .....                                    | 2688        |
| RefreshCurrent.....                              | 2689        |
| CTimeBuffer .....                                | 2690        |
| Size .....                                       | 2691        |
| SetSymbolPeriod.....                             | 2692        |
| At .....                                         | 2693        |
| Refresh .....                                    | 2694        |
| RefreshCurrent.....                              | 2695        |
| CTickVolumeBuffer .....                          | 2696        |
| Size .....                                       | 2697        |
| SetSymbolPeriod.....                             | 2698        |
| At .....                                         | 2699        |
| Refresh .....                                    | 2700        |
| RefreshCurrent.....                              | 2701        |
| CRealVolumeBuffer .....                          | 2702        |
| Size .....                                       | 2703        |
| SetSymbolPeriod.....                             | 2704        |
| At .....                                         | 2705        |
| Refresh .....                                    | 2706        |
| RefreshCurrent.....                              | 2707        |
| CDoubleBuffer.....                               | 2708        |
| Size .....                                       | 2709        |
| SetSymbolPeriod.....                             | 2710        |
| At .....                                         | 2711        |
| Refresh .....                                    | 2712        |

|                        |      |
|------------------------|------|
| RefreshCurrent.....    | 2713 |
| COpenBuffer .....      | 2714 |
| Refresh .....          | 2715 |
| RefreshCurrent.....    | 2716 |
| CHighBuffer .....      | 2717 |
| Refresh .....          | 2718 |
| RefreshCurrent.....    | 2719 |
| CLowBuffer.....        | 2720 |
| Refresh .....          | 2721 |
| RefreshCurrent.....    | 2722 |
| CCloseBuffer .....     | 2723 |
| Refresh .....          | 2724 |
| RefreshCurrent.....    | 2725 |
| CIndicatorBuffer ..... | 2726 |
| Offset .....           | 2727 |
| Name .....             | 2728 |
| At .....               | 2729 |
| Refresh .....          | 2730 |
| RefreshCurrent.....    | 2731 |
| CSeries .....          | 2732 |
| Name .....             | 2733 |
| BuffersTotal.....      | 2734 |
| Timeframe.....         | 2735 |
| Symbol .....           | 2736 |
| Period .....           | 2737 |
| RefreshCurrent.....    | 2738 |
| BufferResize .....     | 2739 |
| Refresh .....          | 2740 |
| PeriodDescription..... | 2741 |
| CPriceSeries.....      | 2742 |
| BufferResize .....     | 2743 |
| GetData .....          | 2744 |
| Refresh .....          | 2745 |
| MinIndex .....         | 2746 |
| MinValue .....         | 2747 |
| MaxIndex .....         | 2748 |
| MaxValue .....         | 2749 |
| CIndicator.....        | 2750 |
| Handle .....           | 2753 |
| Status .....           | 2754 |
| FullRelease.....       | 2755 |
| Create .....           | 2756 |
| BufferResize .....     | 2757 |
| BarsCalculated.....    | 2758 |
| GetData .....          | 2759 |
| Refresh .....          | 2762 |
| Minimum .....          | 2763 |
| MinValue .....         | 2764 |
| Maximum .....          | 2765 |
| MaxValue .....         | 2766 |
| MethodDescription..... | 2767 |
| PriceDescription.....  | 2768 |
| VolumeDescription..... | 2769 |
| AddToChart .....       | 2770 |
| DeleteFromChart.....   | 2771 |
| CIndicators.....       | 2772 |
| Create .....           | 2773 |
| Refresh .....          | 2774 |

|                          |      |
|--------------------------|------|
| Timeseries classes ..... | 2775 |
| CiSpread .....           | 2776 |
| Create .....             | 2777 |
| BufferResize .....       | 2778 |
| GetData .....            | 2779 |
| Refresh .....            | 2781 |
| CiTime .....             | 2782 |
| Create .....             | 2783 |
| BufferResize .....       | 2784 |
| GetData .....            | 2785 |
| Refresh .....            | 2787 |
| CiTickVolume .....       | 2788 |
| Create .....             | 2789 |
| BufferResize .....       | 2790 |
| GetData .....            | 2791 |
| Refresh .....            | 2793 |
| CiRealVolume .....       | 2794 |
| Create .....             | 2795 |
| BufferResize .....       | 2796 |
| GetData .....            | 2797 |
| Refresh .....            | 2799 |
| CiOpen .....             | 2800 |
| Create .....             | 2801 |
| GetData .....            | 2802 |
| CiHigh .....             | 2804 |
| Create .....             | 2805 |
| GetData .....            | 2806 |
| CiLow .....              | 2808 |
| Create .....             | 2809 |
| GetData .....            | 2810 |
| CiClose .....            | 2812 |
| Create .....             | 2813 |
| GetData .....            | 2814 |
| Trend Indicators .....   | 2816 |
| CiADX .....              | 2817 |
| MaPeriod .....           | 2818 |
| Create .....             | 2819 |
| Main .....               | 2820 |
| Plus .....               | 2821 |
| Minus .....              | 2822 |
| Type .....               | 2823 |
| CiADXWilder .....        | 2824 |
| MaPeriod .....           | 2825 |
| Create .....             | 2826 |
| Main .....               | 2827 |
| Plus .....               | 2828 |
| Minus .....              | 2829 |
| Type .....               | 2830 |
| CiBands .....            | 2831 |
| MaPeriod .....           | 2832 |
| MaShift .....            | 2833 |
| Deviation .....          | 2834 |
| Applied .....            | 2835 |
| Create .....             | 2836 |
| Base .....               | 2837 |
| Upper .....              | 2838 |
| Lower .....              | 2839 |
| Type .....               | 2840 |



|                        |      |
|------------------------|------|
| CiEnvelopes.....       | 2841 |
| MaPeriod .....         | 2842 |
| MaShift .....          | 2843 |
| MaMethod.....          | 2844 |
| Deviation .....        | 2845 |
| Applied .....          | 2846 |
| Create .....           | 2847 |
| Upper .....            | 2848 |
| Lower .....            | 2849 |
| Type .....             | 2850 |
| Cilchimoku.....        | 2851 |
| TenkanSenPeriod.....   | 2852 |
| KijunSenPeriod.....    | 2853 |
| SenkouSpanBPeriod..... | 2854 |
| Create .....           | 2855 |
| TenkanSen.....         | 2856 |
| KijunSen .....         | 2857 |
| SenkouSpanA.....       | 2858 |
| SenkouSpanB.....       | 2859 |
| ChinkouSpan.....       | 2860 |
| Type .....             | 2861 |
| CiMA .....             | 2862 |
| MaPeriod .....         | 2863 |
| MaShift .....          | 2864 |
| MaMethod.....          | 2865 |
| Applied .....          | 2866 |
| Create .....           | 2867 |
| Main .....             | 2868 |
| Type .....             | 2869 |
| CiSAR .....            | 2870 |
| SarStep .....          | 2871 |
| Maximum .....          | 2872 |
| Create .....           | 2873 |
| Main .....             | 2874 |
| Type .....             | 2875 |
| CiStdDev .....         | 2876 |
| MaPeriod .....         | 2877 |
| MaShift .....          | 2878 |
| MaMethod.....          | 2879 |
| Applied .....          | 2880 |
| Create .....           | 2881 |
| Main .....             | 2882 |
| Type .....             | 2883 |
| CiDEMA .....           | 2884 |
| MaPeriod .....         | 2885 |
| IndShift .....         | 2886 |
| Applied .....          | 2887 |
| Create .....           | 2888 |
| Main .....             | 2889 |
| Type .....             | 2890 |
| CiTEMA .....           | 2891 |
| MaPeriod .....         | 2892 |
| IndShift .....         | 2893 |
| Applied .....          | 2894 |
| Create .....           | 2895 |
| Main .....             | 2896 |
| Type .....             | 2897 |
| CiFrAMA .....          | 2898 |

|                    |      |
|--------------------|------|
| MaPeriod .....     | 2899 |
| IndShift .....     | 2900 |
| Applied .....      | 2901 |
| Create .....       | 2902 |
| Main .....         | 2903 |
| Type .....         | 2904 |
| CiAMA .....        | 2905 |
| MaPeriod .....     | 2906 |
| FastEmaPeriod..... | 2907 |
| SlowEmaPeriod..... | 2908 |
| IndShift .....     | 2909 |
| Applied .....      | 2910 |
| Create .....       | 2911 |
| Main .....         | 2912 |
| Type .....         | 2913 |
| CiVIDyA .....      | 2914 |
| CmoPeriod.....     | 2915 |
| EmaPeriod.....     | 2916 |
| IndShift .....     | 2917 |
| Applied .....      | 2918 |
| Create .....       | 2919 |
| Main .....         | 2920 |
| Type .....         | 2921 |
| Oscillators.....   | 2922 |
| CiATR .....        | 2923 |
| MaPeriod .....     | 2924 |
| Create .....       | 2925 |
| Main .....         | 2926 |
| Type .....         | 2927 |
| CiBearsPower.....  | 2928 |
| MaPeriod .....     | 2929 |
| Create .....       | 2930 |
| Main .....         | 2931 |
| Type .....         | 2932 |
| CiBullsPower.....  | 2933 |
| MaPeriod .....     | 2934 |
| Create .....       | 2935 |
| Main .....         | 2936 |
| Type .....         | 2937 |
| CiCCI .....        | 2938 |
| MaPeriod .....     | 2939 |
| Applied .....      | 2940 |
| Create .....       | 2941 |
| Main .....         | 2942 |
| Type .....         | 2943 |
| CiChaikin .....    | 2944 |
| FastMaPeriod.....  | 2945 |
| SlowMaPeriod.....  | 2946 |
| MaMethod.....      | 2947 |
| Applied .....      | 2948 |
| Create .....       | 2949 |
| Main .....         | 2950 |
| Type .....         | 2951 |
| CiDeMarker.....    | 2952 |
| MaPeriod .....     | 2953 |
| Create .....       | 2954 |
| Main .....         | 2955 |
| Type .....         | 2956 |

|                     |      |
|---------------------|------|
| CiForce .....       | 2957 |
| MaPeriod .....      | 2958 |
| MaMethod .....      | 2959 |
| Applied .....       | 2960 |
| Create .....        | 2961 |
| Main .....          | 2962 |
| Type .....          | 2963 |
| CiMACD .....        | 2964 |
| FastEmaPeriod ..... | 2965 |
| SlowEmaPeriod ..... | 2966 |
| SignalPeriod .....  | 2967 |
| Applied .....       | 2968 |
| Create .....        | 2969 |
| Main .....          | 2970 |
| Signal .....        | 2971 |
| Type .....          | 2972 |
| CiMomentum .....    | 2973 |
| MaPeriod .....      | 2974 |
| Applied .....       | 2975 |
| Create .....        | 2976 |
| Main .....          | 2977 |
| Type .....          | 2978 |
| CiOsMA .....        | 2979 |
| FastEmaPeriod ..... | 2980 |
| SlowEmaPeriod ..... | 2981 |
| SignalPeriod .....  | 2982 |
| Applied .....       | 2983 |
| Create .....        | 2984 |
| Main .....          | 2985 |
| Type .....          | 2986 |
| CiRSI .....         | 2987 |
| MaPeriod .....      | 2988 |
| Applied .....       | 2989 |
| Create .....        | 2990 |
| Main .....          | 2991 |
| Type .....          | 2992 |
| CiRVI .....         | 2993 |
| MaPeriod .....      | 2994 |
| Create .....        | 2995 |
| Main .....          | 2996 |
| Signal .....        | 2997 |
| Type .....          | 2998 |
| CiStochastic .....  | 2999 |
| Kperiod .....       | 3000 |
| Dperiod .....       | 3001 |
| Slowing .....       | 3002 |
| MaMethod .....      | 3003 |
| PriceField .....    | 3004 |
| Create .....        | 3005 |
| Main .....          | 3006 |
| Signal .....        | 3007 |
| Type .....          | 3008 |
| CiTriX .....        | 3009 |
| MaPeriod .....      | 3010 |
| Applied .....       | 3011 |
| Create .....        | 3012 |
| Main .....          | 3013 |
| Type .....          | 3014 |

|                               |      |
|-------------------------------|------|
| CiWPR .....                   | 3015 |
| CalcPeriod.....               | 3016 |
| Create .....                  | 3017 |
| Main .....                    | 3018 |
| Type .....                    | 3019 |
| Volume Indicators.....        | 3020 |
| CiAD .....                    | 3021 |
| Applied .....                 | 3022 |
| Create .....                  | 3023 |
| Main .....                    | 3024 |
| Type .....                    | 3025 |
| CiMFI .....                   | 3026 |
| MaPeriod .....                | 3027 |
| Applied .....                 | 3028 |
| Create .....                  | 3029 |
| Main .....                    | 3030 |
| Type .....                    | 3031 |
| CiOBV .....                   | 3032 |
| Applied .....                 | 3033 |
| Create .....                  | 3034 |
| Main .....                    | 3035 |
| Type .....                    | 3036 |
| CiVolumes .....               | 3037 |
| Applied .....                 | 3038 |
| Create .....                  | 3039 |
| Main .....                    | 3040 |
| Type .....                    | 3041 |
| Bill Williams Indicators..... | 3042 |
| CiAC .....                    | 3043 |
| Create .....                  | 3044 |
| Main .....                    | 3045 |
| Type .....                    | 3046 |
| CiAlligator .....             | 3047 |
| JawPeriod.....                | 3048 |
| JawShift .....                | 3049 |
| TeethPeriod.....              | 3050 |
| TeethShift .....              | 3051 |
| LipsPeriod.....               | 3052 |
| LipsShift .....               | 3053 |
| MaMethod.....                 | 3054 |
| Applied .....                 | 3055 |
| Create .....                  | 3056 |
| Jaw .....                     | 3057 |
| Teeth .....                   | 3058 |
| Lips .....                    | 3059 |
| Type .....                    | 3060 |
| CiAO .....                    | 3061 |
| Create .....                  | 3062 |
| Main .....                    | 3063 |
| Type .....                    | 3064 |
| CiFractals .....              | 3065 |
| Create .....                  | 3066 |
| Upper .....                   | 3067 |
| Lower .....                   | 3068 |
| Type .....                    | 3069 |
| CiGator .....                 | 3070 |
| JawPeriod.....                | 3071 |
| JawShift .....                | 3072 |

|                            |             |
|----------------------------|-------------|
| TeethPeriod.....           | 3073        |
| TeethShift.....            | 3074        |
| LipsPeriod.....            | 3075        |
| LipsShift .....            | 3076        |
| MaMethod.....              | 3077        |
| Applied .....              | 3078        |
| Create .....               | 3079        |
| Upper .....                | 3080        |
| Lower .....                | 3081        |
| Type .....                 | 3082        |
| CiBWMFI .....              | 3083        |
| Applied .....              | 3084        |
| Create .....               | 3085        |
| Main .....                 | 3086        |
| Type .....                 | 3087        |
| Custom indicators .....    | 3088        |
| NumBuffers.....            | 3089        |
| NumParams.....             | 3090        |
| ParamType.....             | 3091        |
| ParamLong.....             | 3092        |
| ParamDouble.....           | 3093        |
| ParamString.....           | 3094        |
| Type .....                 | 3095        |
| <b>Trade Classes .....</b> | <b>3096</b> |
| CAccountInfo.....          | 3097        |
| Login .....                | 3099        |
| TradeMode.....             | 3100        |
| TradeModeDescription.....  | 3101        |
| Leverage .....             | 3102        |
| MarginMode .....           | 3103        |
| MarginModeDescription..... | 3104        |
| TradeAllowed.....          | 3105        |
| TradeExpert.....           | 3106        |
| LimitOrders .....          | 3107        |
| Balance .....              | 3108        |
| Credit .....               | 3109        |
| Profit .....               | 3110        |
| Equity .....               | 3111        |
| Margin .....               | 3112        |
| FreeMargin.....            | 3113        |
| MarginLevel.....           | 3114        |
| MarginCall .....           | 3115        |
| MarginStopOut.....         | 3116        |
| Name .....                 | 3117        |
| Server .....               | 3118        |
| Currency .....             | 3119        |
| Company .....              | 3120        |
| InfoInteger.....           | 3121        |
| InfoDouble.....            | 3122        |
| InfoString .....           | 3123        |
| OrderProfitCheck.....      | 3124        |
| MarginCheck .....          | 3125        |
| FreeMarginCheck .....      | 3126        |
| MaxLotCheck .....          | 3127        |
| CSymbolInfo.....           | 3128        |
| Refresh .....              | 3132        |
| RefreshRates.....          | 3133        |
| Name .....                 | 3134        |

|                                    |      |
|------------------------------------|------|
| Select .....                       | 3135 |
| IsSynchronized .....               | 3136 |
| Volume .....                       | 3137 |
| VolumeHigh .....                   | 3138 |
| VolumeLow .....                    | 3139 |
| Time .....                         | 3140 |
| Spread .....                       | 3141 |
| SpreadFloat .....                  | 3142 |
| TicksBookDepth .....               | 3143 |
| StopsLevel .....                   | 3144 |
| FreezeLevel .....                  | 3145 |
| Bid .....                          | 3146 |
| BidHigh .....                      | 3147 |
| BidLow .....                       | 3148 |
| Ask .....                          | 3149 |
| AskHigh .....                      | 3150 |
| AskLow .....                       | 3151 |
| Last .....                         | 3152 |
| LastHigh .....                     | 3153 |
| LastLow .....                      | 3154 |
| TradeCalcMode .....                | 3155 |
| TradeCalcModeDescription .....     | 3156 |
| TradeMode .....                    | 3157 |
| TradeModeDescription .....         | 3158 |
| TradeExecution .....               | 3159 |
| TradeExecutionDescription .....    | 3160 |
| SwapMode .....                     | 3161 |
| SwapModeDescription .....          | 3162 |
| SwapRollover3days .....            | 3163 |
| SwapRollover3daysDescription ..... | 3164 |
| MarginInitial .....                | 3165 |
| MarginMaintenance .....            | 3166 |
| MarginLong .....                   | 3167 |
| MarginShort .....                  | 3168 |
| MarginLimit .....                  | 3169 |
| MarginStop .....                   | 3170 |
| MarginStopLimit .....              | 3171 |
| TradeTimeFlags .....               | 3172 |
| TradeFillFlags .....               | 3173 |
| Digits .....                       | 3174 |
| Point .....                        | 3175 |
| TickValue .....                    | 3176 |
| TickValueProfit .....              | 3177 |
| TickValueLoss .....                | 3178 |
| TickSize .....                     | 3179 |
| ContractSize .....                 | 3180 |
| LotsMin .....                      | 3181 |
| LotsMax .....                      | 3182 |
| LotsStep .....                     | 3183 |
| LotsLimit .....                    | 3184 |
| SwapLong .....                     | 3185 |
| SwapShort .....                    | 3186 |
| CurrencyBase .....                 | 3187 |
| CurrencyProfit .....               | 3188 |
| CurrencyMargin .....               | 3189 |
| Bank .....                         | 3190 |
| Description .....                  | 3191 |
| Path .....                         | 3192 |

|                              |      |
|------------------------------|------|
| SessionDeals.....            | 3193 |
| SessionBuyOrders.....        | 3194 |
| SessionSellOrders.....       | 3195 |
| SessionTurnover.....         | 3196 |
| SessionInterest.....         | 3197 |
| SessionBuyOrdersVolume.....  | 3198 |
| SessionSellOrdersVolume..... | 3199 |
| SessionOpen.....             | 3200 |
| SessionClose.....            | 3201 |
| SessionAW.....               | 3202 |
| SessionPriceSettlement.....  | 3203 |
| SessionPriceLimitMin.....    | 3204 |
| SessionPriceLimitMax.....    | 3205 |
| InfoInteger.....             | 3206 |
| InfoDouble.....              | 3207 |
| InfoString.....              | 3208 |
| NormalizePrice.....          | 3209 |
| COrderInfo.....              | 3210 |
| Ticket.....                  | 3212 |
| TimeSetup.....               | 3213 |
| TimeSetupMsc.....            | 3214 |
| OrderType.....               | 3215 |
| TypeDescription.....         | 3216 |
| State.....                   | 3217 |
| StateDescription.....        | 3218 |
| TimeExpiration.....          | 3219 |
| TimeDone.....                | 3220 |
| TimeDoneMsc.....             | 3221 |
| TypeFilling.....             | 3222 |
| TypeFillingDescription.....  | 3223 |
| TypeTime.....                | 3224 |
| TypeTimeDescription.....     | 3225 |
| Magic.....                   | 3226 |
| PositionId.....              | 3227 |
| VolumeInitial.....           | 3228 |
| VolumeCurrent.....           | 3229 |
| PriceOpen.....               | 3230 |
| StopLoss.....                | 3231 |
| TakeProfit.....              | 3232 |
| PriceCurrent.....            | 3233 |
| PriceStopLimit.....          | 3234 |
| Symbol.....                  | 3235 |
| Comment.....                 | 3236 |
| InfoInteger.....             | 3237 |
| InfoDouble.....              | 3238 |
| InfoString.....              | 3239 |
| StoreState.....              | 3240 |
| CheckState.....              | 3241 |
| Select.....                  | 3242 |
| SelectByIndex.....           | 3243 |
| CHistoryOrderInfo.....       | 3244 |
| TimeSetup.....               | 3246 |
| TimeSetupMsc.....            | 3247 |
| OrderType.....               | 3248 |
| TypeDescription.....         | 3249 |
| State.....                   | 3250 |
| StateDescription.....        | 3251 |
| TimeExpiration.....          | 3252 |

|                             |      |
|-----------------------------|------|
| TimeDone .....              | 3253 |
| TimeDoneMsc.....            | 3254 |
| TypeFilling .....           | 3255 |
| TypeFillingDescription..... | 3256 |
| TypeTime .....              | 3257 |
| TypeTimeDescription.....    | 3258 |
| Magic .....                 | 3259 |
| PositionId .....            | 3260 |
| VolumeInitial.....          | 3261 |
| VolumeCurrent.....          | 3262 |
| PriceOpen .....             | 3263 |
| StopLoss .....              | 3264 |
| TakeProfit .....            | 3265 |
| PriceCurrent.....           | 3266 |
| PriceStopLimit .....        | 3267 |
| Symbol .....                | 3268 |
| Comment .....               | 3269 |
| InfoInteger.....            | 3270 |
| InfoDouble.....             | 3271 |
| InfoString .....            | 3272 |
| Ticket .....                | 3273 |
| SelectByIndex.....          | 3274 |
| CPositionInfo .....         | 3275 |
| Time .....                  | 3277 |
| TimeMsc .....               | 3278 |
| TimeUpdate.....             | 3279 |
| TimeUpdateMsc.....          | 3280 |
| PositionType .....          | 3281 |
| TypeDescription.....        | 3282 |
| Magic .....                 | 3283 |
| Identifier .....            | 3284 |
| Volume .....                | 3285 |
| PriceOpen .....             | 3286 |
| StopLoss .....              | 3287 |
| TakeProfit .....            | 3288 |
| PriceCurrent.....           | 3289 |
| Commission.....             | 3290 |
| Swap .....                  | 3291 |
| Profit .....                | 3292 |
| Symbol .....                | 3293 |
| Comment .....               | 3294 |
| InfoInteger.....            | 3295 |
| InfoDouble.....             | 3296 |
| InfoString .....            | 3297 |
| Select .....                | 3298 |
| SelectByIndex.....          | 3299 |
| StoreState.....             | 3300 |
| CheckState.....             | 3301 |
| CDealInfo .....             | 3302 |
| Order .....                 | 3304 |
| Time .....                  | 3305 |
| TimeMsc .....               | 3306 |
| DealType .....              | 3307 |
| TypeDescription.....        | 3308 |
| Entry .....                 | 3309 |
| EntryDescription.....       | 3310 |
| Magic .....                 | 3311 |
| PositionId .....            | 3312 |



|                                     |      |
|-------------------------------------|------|
| Volume .....                        | 3313 |
| Price .....                         | 3314 |
| Commision .....                     | 3315 |
| Swap .....                          | 3316 |
| Profit .....                        | 3317 |
| Symbol .....                        | 3318 |
| Comment .....                       | 3319 |
| InfoInteger .....                   | 3320 |
| InfoDouble .....                    | 3321 |
| InfoString .....                    | 3322 |
| Ticket .....                        | 3323 |
| SelectByIndex .....                 | 3324 |
| CTrade .....                        | 3325 |
| LogLevel .....                      | 3329 |
| SetExpertMagicNumber .....          | 3330 |
| SetDeviationInPoints .....          | 3331 |
| SetTypeFilling .....                | 3332 |
| SetAsyncMode .....                  | 3333 |
| OrderOpen .....                     | 3334 |
| OrderModify .....                   | 3336 |
| OrderDelete .....                   | 3337 |
| PositionOpen .....                  | 3338 |
| PositionModify .....                | 3339 |
| PositionClose .....                 | 3340 |
| Buy .....                           | 3341 |
| Sell .....                          | 3342 |
| BuyLimit .....                      | 3343 |
| BuyStop .....                       | 3344 |
| SellLimit .....                     | 3345 |
| SellStop .....                      | 3346 |
| Request .....                       | 3347 |
| RequestAction .....                 | 3348 |
| RequestActionDescription .....      | 3349 |
| RequestMagic .....                  | 3350 |
| RequestOrder .....                  | 3351 |
| RequestSymbol .....                 | 3352 |
| RequestVolume .....                 | 3353 |
| RequestPrice .....                  | 3354 |
| RequestStopLimit .....              | 3355 |
| RequestSL .....                     | 3356 |
| RequestTP .....                     | 3357 |
| RequestDeviation .....              | 3358 |
| RequestType .....                   | 3359 |
| RequestTypeDescription .....        | 3360 |
| RequestTypeFilling .....            | 3361 |
| RequestTypeFillingDescription ..... | 3362 |
| RequestTypeTime .....               | 3363 |
| RequestTypeTimeDescription .....    | 3364 |
| RequestExpiration .....             | 3365 |
| RequestComment .....                | 3366 |
| Result .....                        | 3367 |
| ResultRetcode .....                 | 3368 |
| ResultRetcodeDescription .....      | 3369 |
| ResultDeal .....                    | 3370 |
| ResultOrder .....                   | 3371 |
| ResultVolume .....                  | 3372 |
| ResultPrice .....                   | 3373 |
| ResultBid .....                     | 3374 |

|                                        |             |
|----------------------------------------|-------------|
| ResultAsk .....                        | 3375        |
| ResultComment .....                    | 3376        |
| CheckResult .....                      | 3377        |
| CheckResultRetcode .....               | 3378        |
| CheckResultRetcodeDescription .....    | 3379        |
| CheckResultBalance .....               | 3380        |
| CheckResultEquity .....                | 3381        |
| CheckResultProfit .....                | 3382        |
| CheckResultMargin .....                | 3383        |
| CheckResultMarginFree .....            | 3384        |
| CheckResultMarginLevel .....           | 3385        |
| CheckResultComment .....               | 3386        |
| PrintRequest .....                     | 3387        |
| PrintResult .....                      | 3388        |
| FormatRequest .....                    | 3389        |
| FormatRequestResult .....              | 3390        |
| CTerminalInfo .....                    | 3391        |
| Build .....                            | 3393        |
| IsConnected .....                      | 3394        |
| IsDLLsAllowed .....                    | 3395        |
| IsTradeAllowed .....                   | 3396        |
| IsEmailEnabled .....                   | 3397        |
| IsFtpEnabled .....                     | 3398        |
| MaxBars .....                          | 3399        |
| CodePage .....                         | 3400        |
| CPUCores .....                         | 3401        |
| MemoryPhysical .....                   | 3402        |
| MemoryTotal .....                      | 3403        |
| MemoryAvailable .....                  | 3404        |
| MemoryUsed .....                       | 3405        |
| IsX64 .....                            | 3406        |
| OpenCLSupport .....                    | 3407        |
| DiskSpace .....                        | 3408        |
| Language .....                         | 3409        |
| Name .....                             | 3410        |
| Company .....                          | 3411        |
| Path .....                             | 3412        |
| DataPath .....                         | 3413        |
| CommonDataPath .....                   | 3414        |
| InfoInteger .....                      | 3415        |
| InfoString .....                       | 3416        |
| <b>Trading Strategy Classes .....</b>  | <b>3417</b> |
| Base classes for Expert Advisors ..... | 3420        |
| CExpertBase .....                      | 3421        |
| InitPhase .....                        | 3423        |
| TrendType .....                        | 3424        |
| UsedSeries .....                       | 3425        |
| EveryTick .....                        | 3426        |
| Open .....                             | 3427        |
| High .....                             | 3428        |
| Low .....                              | 3429        |
| Close .....                            | 3430        |
| Spread .....                           | 3431        |
| Time .....                             | 3432        |
| TickVolume .....                       | 3433        |
| RealVolume .....                       | 3434        |
| Init .....                             | 3435        |
| Symbol .....                           | 3436        |

|                           |      |
|---------------------------|------|
| Period .....              | 3437 |
| Magic .....               | 3438 |
| ValidationSettings .....  | 3439 |
| SetPriceSeries .....      | 3440 |
| SetOtherSeries .....      | 3441 |
| InitIndicators .....      | 3442 |
| InitOpen .....            | 3443 |
| InitHigh .....            | 3444 |
| InitLow .....             | 3445 |
| InitClose .....           | 3446 |
| InitSpread .....          | 3447 |
| InitTime .....            | 3448 |
| InitTickVolume .....      | 3449 |
| InitRealVolume .....      | 3450 |
| PriceLevelUnit .....      | 3451 |
| StartIndex .....          | 3452 |
| CompareMagic .....        | 3453 |
| CExpert .....             | 3454 |
| Init .....                | 3458 |
| Magic .....               | 3459 |
| InitSignal .....          | 3460 |
| InitTrailing .....        | 3461 |
| InitMoney .....           | 3462 |
| InitTrade .....           | 3463 |
| Deinit .....              | 3464 |
| OnTickProcess .....       | 3465 |
| OnTradeProcess .....      | 3466 |
| OnTimerProcess .....      | 3467 |
| OnChartEventProcess ..... | 3468 |
| OnBookEventProcess .....  | 3469 |
| MaxOrders .....           | 3470 |
| Signal .....              | 3471 |
| ValidationSettings .....  | 3472 |
| InitIndicators .....      | 3473 |
| OnTick .....              | 3474 |
| OnTrade .....             | 3475 |
| OnTimer .....             | 3476 |
| OnChartEvent .....        | 3477 |
| OnBookEvent .....         | 3478 |
| InitParameters .....      | 3479 |
| DeinitTrade .....         | 3480 |
| DeinitSignal .....        | 3481 |
| DeinitTrailing .....      | 3482 |
| DeinitMoney .....         | 3483 |
| DeinitIndicators .....    | 3484 |
| Refresh .....             | 3485 |
| Processing .....          | 3486 |
| CheckOpen .....           | 3488 |
| CheckOpenLong .....       | 3489 |
| CheckOpenShort .....      | 3490 |
| OpenLong .....            | 3491 |
| OpenShort .....           | 3492 |
| CheckReverse .....        | 3493 |
| CheckReverseLong .....    | 3494 |
| CheckReverseShort .....   | 3495 |
| ReverseLong .....         | 3496 |
| ReverseShort .....        | 3497 |
| CheckClose .....          | 3498 |

|                                      |      |
|--------------------------------------|------|
| CheckCloseLong.....                  | 3499 |
| CheckCloseShort.....                 | 3500 |
| CloseAll.....                        | 3501 |
| Close.....                           | 3502 |
| CloseLong.....                       | 3503 |
| CloseShort.....                      | 3504 |
| CheckTrailingStop.....               | 3505 |
| CheckTrailingStopLong.....           | 3506 |
| CheckTrailingStopShort.....          | 3507 |
| TrailingStopLong.....                | 3508 |
| TrailingStopShort.....               | 3509 |
| CheckTrailingOrderLong.....          | 3510 |
| CheckTrailingOrderShort.....         | 3511 |
| TrailingOrderLong.....               | 3512 |
| TrailingOrderShort.....              | 3513 |
| CheckDeleteOrderLong.....            | 3514 |
| CheckDeleteOrderShort.....           | 3515 |
| DeleteOrders.....                    | 3516 |
| DeleteOrder.....                     | 3517 |
| DeleteOrderLong.....                 | 3518 |
| DeleteOrderShort.....                | 3519 |
| LotOpenLong.....                     | 3520 |
| LotOpenShort.....                    | 3521 |
| LotReverse.....                      | 3522 |
| PrepareHistoryDate.....              | 3523 |
| HistoryPoint.....                    | 3524 |
| CheckTradeState.....                 | 3525 |
| WaitEvent.....                       | 3526 |
| NoWaitEvent.....                     | 3527 |
| TradeEventPositionStopTake.....      | 3528 |
| TradeEventOrderTriggered.....        | 3529 |
| TradeEventPositionOpened.....        | 3530 |
| TradeEventPositionVolumeChanged..... | 3531 |
| TradeEventPositionModified.....      | 3532 |
| TradeEventPositionClosed.....        | 3533 |
| TradeEventOrderPlaced.....           | 3534 |
| TradeEventOrderModified.....         | 3535 |
| TradeEventOrderDeleted.....          | 3536 |
| TradeEventNotIdentified.....         | 3537 |
| TimeframeAdd.....                    | 3538 |
| TimeframesFlags.....                 | 3539 |
| CExpertSignal.....                   | 3540 |
| BasePrice.....                       | 3542 |
| UsedSeries.....                      | 3543 |
| Weight.....                          | 3544 |
| PatternsUsage.....                   | 3545 |
| General.....                         | 3546 |
| Ignore.....                          | 3547 |
| Invert.....                          | 3548 |
| ThresholdOpen.....                   | 3549 |
| ThresholdClose.....                  | 3550 |
| PriceLevel.....                      | 3551 |
| StopLevel.....                       | 3552 |
| TakeLevel.....                       | 3553 |
| Expiration.....                      | 3554 |
| Magic.....                           | 3555 |
| ValidationSettings.....              | 3556 |
| InitIndicators.....                  | 3557 |

|                                                                 |      |
|-----------------------------------------------------------------|------|
| AddFilter .....                                                 | 3558 |
| CheckOpenLong.....                                              | 3559 |
| CheckOpenShort .....                                            | 3560 |
| OpenLongParams.....                                             | 3561 |
| OpenShortParams.....                                            | 3562 |
| CheckCloseLong.....                                             | 3563 |
| CheckCloseShort .....                                           | 3564 |
| CloseLongParams.....                                            | 3565 |
| CloseShortParams.....                                           | 3566 |
| CheckReverseLong .....                                          | 3567 |
| CheckReverseShort.....                                          | 3568 |
| CheckTrailingOrderLong.....                                     | 3569 |
| CheckTrailingOrderShort.....                                    | 3570 |
| LongCondition .....                                             | 3571 |
| ShortCondition .....                                            | 3572 |
| Direction .....                                                 | 3573 |
| CExpertTrailing.....                                            | 3574 |
| CheckTrailingStopLong.....                                      | 3575 |
| CheckTrailingStopShort.....                                     | 3576 |
| CExpertMoney .....                                              | 3577 |
| Percent .....                                                   | 3578 |
| ValidationSettings .....                                        | 3579 |
| CheckOpenLong.....                                              | 3580 |
| CheckOpenShort .....                                            | 3581 |
| CheckReverse .....                                              | 3582 |
| CheckClose.....                                                 | 3583 |
| Modules of Trade Signals .....                                  | 3584 |
| Signals of the Indicator Accelerator Oscillator .....           | 3587 |
| Signals of the Indicator Adaptive Moving Average.....           | 3590 |
| Signals of the Indicator Awesome Oscillator.....                | 3594 |
| Signals of the Oscillator Bears Power.....                      | 3598 |
| Signals of the Oscillator Bulls Power.....                      | 3600 |
| Signals of the Oscillator Commodity Channel Index .....         | 3602 |
| Signals of the Oscillator DeMarker .....                        | 3606 |
| Signals of the Indicator Double Exponential Moving Average..... | 3610 |
| Signals of the Indicator Envelopes .....                        | 3614 |
| Signals of the Indicator Fractal Adaptive Moving Average .....  | 3617 |
| Signals of the Intraday Time Filter .....                       | 3621 |
| Signals of the Oscillator MACD .....                            | 3623 |
| Signals of the Indicator Moving Average.....                    | 3629 |
| Signals of the Indicator Parabolic SAR.....                     | 3633 |
| Signals of the Oscillator Relative Strength Index.....          | 3635 |
| Signals of the Oscillator Relative Vigor Index .....            | 3641 |
| Signals of the Oscillator Stochastic .....                      | 3643 |
| Signals of the Oscillator Triple Exponential Average.....       | 3648 |
| Signals of the Indicator Triple Exponential Moving Average..... | 3652 |
| Signals of the Oscillator Williams Percent Range.....           | 3656 |
| Trailing Stop Classes.....                                      | 3659 |
| CTrailingFixedPips.....                                         | 3660 |
| StopLevel .....                                                 | 3661 |
| ProfitLevel.....                                                | 3662 |
| ValidationSettings .....                                        | 3663 |
| CheckTrailingStopLong.....                                      | 3664 |
| CheckTrailingStopShort.....                                     | 3665 |
| CTrailingMA.....                                                | 3666 |
| Period .....                                                    | 3667 |
| Shift .....                                                     | 3668 |
| Method .....                                                    | 3669 |

|                                                     |             |
|-----------------------------------------------------|-------------|
| Applied .....                                       | 3670        |
| InitIndicators.....                                 | 3671        |
| ValidationSettings .....                            | 3672        |
| CheckTrailingStopLong.....                          | 3673        |
| CheckTrailingStopShort.....                         | 3674        |
| CTrailingNone.....                                  | 3675        |
| CheckTrailingStopLong.....                          | 3676        |
| CheckTrailingStopShort.....                         | 3677        |
| CTrailingPSAR.....                                  | 3678        |
| Step .....                                          | 3679        |
| Maximum .....                                       | 3680        |
| InitIndicators.....                                 | 3681        |
| CheckTrailingStopLong.....                          | 3682        |
| CheckTrailingStopShort.....                         | 3683        |
| Money Management Classes.....                       | 3684        |
| CMoneyFixedLot.....                                 | 3685        |
| Lots .....                                          | 3686        |
| ValidationSettings .....                            | 3687        |
| CheckOpenLong.....                                  | 3688        |
| CheckOpenShort .....                                | 3689        |
| CMoneyFixedMargin .....                             | 3690        |
| CheckOpenLong.....                                  | 3691        |
| CheckOpenShort .....                                | 3692        |
| CMoneyFixedRisk.....                                | 3693        |
| CheckOpenLong.....                                  | 3694        |
| CheckOpenShort .....                                | 3695        |
| CMoneyNone .....                                    | 3696        |
| ValidationSettings .....                            | 3697        |
| CheckOpenLong.....                                  | 3698        |
| CheckOpenShort .....                                | 3699        |
| CMoneySizeOptimized.....                            | 3700        |
| DecreaseFactor.....                                 | 3701        |
| ValidationSettings .....                            | 3702        |
| CheckOpenLong.....                                  | 3703        |
| CheckOpenShort .....                                | 3704        |
| <b>Classes for Control Panels and Dialogs .....</b> | <b>3705</b> |
| CPoint .....                                        | 3707        |
| Move .....                                          | 3708        |
| Shift .....                                         | 3709        |
| Format .....                                        | 3710        |
| CRect .....                                         | 3711        |
| Left .....                                          | 3712        |
| Top .....                                           | 3713        |
| Right .....                                         | 3714        |
| Bottom .....                                        | 3715        |
| Width .....                                         | 3716        |
| Height .....                                        | 3717        |
| SetBound .....                                      | 3718        |
| Move .....                                          | 3719        |
| Shift .....                                         | 3720        |
| Contains .....                                      | 3721        |
| Format .....                                        | 3722        |
| CWnd .....                                          | 3723        |
| Create .....                                        | 3726        |
| Destroy .....                                       | 3727        |
| OnEvent .....                                       | 3728        |
| OnMouseEvent.....                                   | 3729        |
| Name .....                                          | 3730        |

|                         |      |
|-------------------------|------|
| ControlsTotal .....     | 3731 |
| Control .....           | 3732 |
| ControlFind .....       | 3733 |
| Rect .....              | 3734 |
| Left .....              | 3735 |
| Top .....               | 3736 |
| Right .....             | 3737 |
| Bottom .....            | 3738 |
| Width .....             | 3739 |
| Height .....            | 3740 |
| Move .....              | 3741 |
| Shift .....             | 3742 |
| Resize .....            | 3743 |
| Contains .....          | 3744 |
| Alignment .....         | 3745 |
| Align .....             | 3746 |
| Id .....                | 3747 |
| IsEnabled .....         | 3748 |
| Enable .....            | 3749 |
| Disable .....           | 3750 |
| IsVisible .....         | 3751 |
| Visible .....           | 3752 |
| Show .....              | 3753 |
| Hide .....              | 3754 |
| IsActive .....          | 3755 |
| Activate .....          | 3756 |
| Deactivate .....        | 3757 |
| StateFlags .....        | 3758 |
| StateFlagsSet .....     | 3759 |
| StateFlagsReset .....   | 3760 |
| PropFlags .....         | 3761 |
| PropFlagsSet .....      | 3762 |
| PropFlagsReset .....    | 3763 |
| MouseX .....            | 3764 |
| MouseY .....            | 3765 |
| MouseFlags .....        | 3766 |
| MouseFocusKill .....    | 3767 |
| OnCreate .....          | 3768 |
| OnDestroy .....         | 3769 |
| OnMove .....            | 3770 |
| OnResize .....          | 3771 |
| OnEnable .....          | 3772 |
| OnDisable .....         | 3773 |
| OnShow .....            | 3774 |
| OnHide .....            | 3775 |
| OnActivate .....        | 3776 |
| OnDeactivate .....      | 3777 |
| OnClick .....           | 3778 |
| OnChange .....          | 3779 |
| OnMouseDown .....       | 3780 |
| OnMouseUp .....         | 3781 |
| OnDragStart .....       | 3782 |
| OnDragProcess .....     | 3783 |
| OnDragEnd .....         | 3784 |
| DragObjectCreate .....  | 3785 |
| DragObjectDestroy ..... | 3786 |
| CWndContainer .....     | 3787 |
| Destroy .....           | 3789 |

|                           |      |
|---------------------------|------|
| OnEvent .....             | 3790 |
| OnMouseEvent.....         | 3791 |
| ControlsTotal.....        | 3792 |
| Control .....             | 3793 |
| ControlFind.....          | 3794 |
| Add .....                 | 3795 |
| Delete .....              | 3796 |
| Move .....                | 3797 |
| Shift .....               | 3798 |
| Id .....                  | 3799 |
| Enable .....              | 3800 |
| Disable .....             | 3801 |
| Show .....                | 3802 |
| Hide .....                | 3803 |
| MouseFocusKill.....       | 3804 |
| Save .....                | 3805 |
| Load .....                | 3806 |
| OnResize .....            | 3807 |
| OnActivate.....           | 3808 |
| OnDeactivate.....         | 3809 |
| CWndObj .....             | 3810 |
| OnEvent .....             | 3812 |
| Text .....                | 3813 |
| Color .....               | 3814 |
| ColorBackground.....      | 3815 |
| ColorBorder .....         | 3816 |
| Font .....                | 3817 |
| FontSize .....            | 3818 |
| ZOrder .....              | 3819 |
| OnObjectCreate .....      | 3820 |
| OnObjectChange.....       | 3821 |
| OnObjectDelete.....       | 3822 |
| OnObjectDrag.....         | 3823 |
| OnSetText .....           | 3824 |
| OnSetColor .....          | 3825 |
| OnSetColorBackground..... | 3826 |
| OnSetFont .....           | 3827 |
| OnSetFontSize.....        | 3828 |
| OnSetZOrder.....          | 3829 |
| OnDestroy.....            | 3830 |
| OnChange .....            | 3831 |
| CLabel .....              | 3832 |
| Create .....              | 3833 |
| OnSetText .....           | 3834 |
| OnSetColor .....          | 3835 |
| OnSetFont .....           | 3836 |
| OnSetFontSize.....        | 3837 |
| OnCreate .....            | 3838 |
| OnShow .....              | 3839 |
| OnHide .....              | 3840 |
| OnMove .....              | 3841 |
| CBmpButton.....           | 3842 |
| Create .....              | 3844 |
| Border .....              | 3845 |
| BmpNames .....            | 3846 |
| BmpOffName.....           | 3847 |
| BmpOnName.....            | 3848 |
| BmpPassiveName .....      | 3849 |



|                           |      |
|---------------------------|------|
| BmpActiveName.....        | 3850 |
| Pressed .....             | 3851 |
| Locking .....             | 3852 |
| OnSetZOrder.....          | 3853 |
| OnCreate .....            | 3854 |
| OnShow .....              | 3855 |
| OnHide .....              | 3856 |
| OnMove .....              | 3857 |
| OnChange .....            | 3858 |
| OnActivate.....           | 3859 |
| OnDeactivate.....         | 3860 |
| OnMouseDown.....          | 3861 |
| OnMouseUp.....            | 3862 |
| CButton.....              | 3863 |
| Create .....              | 3865 |
| Pressed .....             | 3866 |
| Locking .....             | 3867 |
| OnSetText.....            | 3868 |
| OnSetColor .....          | 3869 |
| OnSetColorBackground..... | 3870 |
| OnSetColorBorder.....     | 3871 |
| OnSetFont .....           | 3872 |
| OnSetFontSize.....        | 3873 |
| OnCreate .....            | 3874 |
| OnShow .....              | 3875 |
| OnHide .....              | 3876 |
| OnMove .....              | 3877 |
| OnResize .....            | 3878 |
| OnMouseDown .....         | 3879 |
| OnMouseUp.....            | 3880 |
| CEdit .....               | 3881 |
| Create .....              | 3883 |
| ReadOnly .....            | 3884 |
| TextAlign .....           | 3885 |
| OnObjectEndEdit.....      | 3886 |
| OnSetText .....           | 3887 |
| OnSetColor .....          | 3888 |
| OnSetColorBackground..... | 3889 |
| OnSetColorBorder.....     | 3890 |
| OnSetFont .....           | 3891 |
| OnSetFontSize.....        | 3892 |
| OnSetZOrder.....          | 3893 |
| OnCreate .....            | 3894 |
| OnShow .....              | 3895 |
| OnHide .....              | 3896 |
| OnMove .....              | 3897 |
| OnResize .....            | 3898 |
| OnChange .....            | 3899 |
| OnClick .....             | 3900 |
| CPanel .....              | 3901 |
| Create .....              | 3902 |
| BorderType.....           | 3903 |
| OnSetText .....           | 3904 |
| OnSetColorBackground..... | 3905 |
| OnSetColorBorder.....     | 3906 |
| OnCreate .....            | 3907 |
| OnShow .....              | 3908 |
| OnHide .....              | 3909 |

|                          |      |
|--------------------------|------|
| OnMove .....             | 3910 |
| OnResize .....           | 3911 |
| OnChange .....           | 3912 |
| CPicture.....            | 3913 |
| Create .....             | 3914 |
| Border .....             | 3915 |
| BmpName .....            | 3916 |
| OnCreate .....           | 3917 |
| OnShow .....             | 3918 |
| OnHide .....             | 3919 |
| OnMove .....             | 3920 |
| OnChange .....           | 3921 |
| CScroll .....            | 3922 |
| Create .....             | 3924 |
| OnEvent .....            | 3925 |
| MinPos .....             | 3926 |
| MaxPos .....             | 3927 |
| CurrPos .....            | 3928 |
| CreateBack.....          | 3929 |
| CreateInc .....          | 3930 |
| CreateDec.....           | 3931 |
| CreateThumb.....         | 3932 |
| OnClickInc.....          | 3933 |
| OnClickDec.....          | 3934 |
| OnShow .....             | 3935 |
| OnHide .....             | 3936 |
| OnChangePos .....        | 3937 |
| OnThumbDragStart .....   | 3938 |
| OnThumbDragProcess ..... | 3939 |
| OnThumbDragEnd .....     | 3940 |
| CalcPos .....            | 3941 |
| CScrollV .....           | 3942 |
| CreateInc .....          | 3943 |
| CreateDec.....           | 3944 |
| CreateThumb.....         | 3945 |
| OnResize .....           | 3946 |
| OnChangePos .....        | 3947 |
| OnThumbDragStart .....   | 3948 |
| OnThumbDragProcess ..... | 3949 |
| OnThumbDragEnd .....     | 3950 |
| CalcPos .....            | 3951 |
| CScrollH.....            | 3952 |
| CreateInc .....          | 3953 |
| CreateDec.....           | 3954 |
| CreateThumb.....         | 3955 |
| OnResize .....           | 3956 |
| OnChangePos .....        | 3957 |
| OnThumbDragStart .....   | 3958 |
| OnThumbDragProcess ..... | 3959 |
| OnThumbDragEnd .....     | 3960 |
| CalcPos .....            | 3961 |
| CWndClient.....          | 3962 |
| Create .....             | 3964 |
| OnEvent .....            | 3965 |
| ColorBackground.....     | 3966 |
| ColorBorder .....        | 3967 |
| BorderType.....          | 3968 |
| VScrolled .....          | 3969 |

|                        |      |
|------------------------|------|
| HScrolled .....        | 3970 |
| CreateBack.....        | 3971 |
| CreateScrollV .....    | 3972 |
| CreateScrollH.....     | 3973 |
| OnResize .....         | 3974 |
| OnVScrollShow.....     | 3975 |
| OnVScrollHide.....     | 3976 |
| OnHScrollShow.....     | 3977 |
| OnHScrollHide.....     | 3978 |
| OnScrollLineDown.....  | 3979 |
| OnScrollLineUp.....    | 3980 |
| OnScrollLineLeft.....  | 3981 |
| OnScrollLineRight..... | 3982 |
| Rebound .....          | 3983 |
| CListView.....         | 3984 |
| Create .....           | 3986 |
| OnEvent .....          | 3987 |
| TotalView .....        | 3988 |
| AddItem .....          | 3989 |
| Select .....           | 3990 |
| SelectByText.....      | 3991 |
| SelectByValue.....     | 3992 |
| Value .....            | 3993 |
| CreateRow.....         | 3994 |
| OnResize .....         | 3995 |
| OnVScrollShow.....     | 3996 |
| OnVScrollHide.....     | 3997 |
| OnScrollLineDown.....  | 3998 |
| OnScrollLineUp.....    | 3999 |
| OnItemClick.....       | 4000 |
| Redraw .....           | 4001 |
| RowState .....         | 4002 |
| CheckView.....         | 4003 |
| CComboBox.....         | 4004 |
| Create .....           | 4006 |
| OnEvent .....          | 4007 |
| AddItem .....          | 4008 |
| ListViewItems.....     | 4009 |
| Select .....           | 4010 |
| SelectByText.....      | 4011 |
| SelectByValue.....     | 4012 |
| Value .....            | 4013 |
| CreateEdit.....        | 4014 |
| CreateButton.....      | 4015 |
| CreateList.....        | 4016 |
| OnClickEdit.....       | 4017 |
| OnClickButton.....     | 4018 |
| OnChangeList.....      | 4019 |
| ListShow .....         | 4020 |
| ListHide .....         | 4021 |
| CCheckBox.....         | 4022 |
| Create .....           | 4023 |
| OnEvent .....          | 4024 |
| Text .....             | 4025 |
| Color .....            | 4026 |
| Checked .....          | 4027 |
| Value .....            | 4028 |
| CreateButton.....      | 4029 |

|                        |      |
|------------------------|------|
| CreateLabel.....       | 4030 |
| OnClickButton.....     | 4031 |
| OnClickLabel.....      | 4032 |
| CCheckGroup.....       | 4033 |
| Create.....            | 4035 |
| OnEvent.....           | 4036 |
| AddItem.....           | 4037 |
| Value.....             | 4038 |
| CreateButton.....      | 4039 |
| OnVScrollShow.....     | 4040 |
| OnVScrollHide.....     | 4041 |
| OnScrollLineDown.....  | 4042 |
| OnScrollLineUp.....    | 4043 |
| OnChangeItem.....      | 4044 |
| Redraw.....            | 4045 |
| RowState.....          | 4046 |
| CRadioButton.....      | 4047 |
| Create.....            | 4048 |
| OnEvent.....           | 4049 |
| Text.....              | 4050 |
| Color.....             | 4051 |
| State.....             | 4052 |
| CreateButton.....      | 4053 |
| CreateLabel.....       | 4054 |
| OnClickButton.....     | 4055 |
| OnClickLabel.....      | 4056 |
| CRadioGroup.....       | 4057 |
| Create.....            | 4059 |
| OnEvent.....           | 4060 |
| AddItem.....           | 4061 |
| Value.....             | 4062 |
| CreateButton.....      | 4063 |
| OnVScrollShow.....     | 4064 |
| OnVScrollHide.....     | 4065 |
| OnScrollLineDown.....  | 4066 |
| OnScrollLineUp.....    | 4067 |
| OnChangeItem.....      | 4068 |
| Redraw.....            | 4069 |
| RowState.....          | 4070 |
| Select.....            | 4071 |
| CSpinEdit.....         | 4072 |
| Create.....            | 4073 |
| OnEvent.....           | 4074 |
| MinValue.....          | 4075 |
| MaxValue.....          | 4076 |
| Value.....             | 4077 |
| CreateEdit.....        | 4078 |
| CreateInc.....         | 4079 |
| CreateDec.....         | 4080 |
| OnClickInc.....        | 4081 |
| OnClickDec.....        | 4082 |
| OnChangeValue.....     | 4083 |
| CDialog.....           | 4084 |
| Create.....            | 4086 |
| OnEvent.....           | 4087 |
| Caption.....           | 4088 |
| Add.....               | 4089 |
| CreateWhiteBorder..... | 4090 |

|                                  |             |
|----------------------------------|-------------|
| CreateBackground.....            | 4091        |
| CreateCaption.....               | 4092        |
| CreateButtonClose.....           | 4093        |
| CreateClientArea.....            | 4094        |
| OnClickCaption.....              | 4095        |
| OnClickButtonClose.....          | 4096        |
| ClientAreaVisible.....           | 4097        |
| ClientAreaLeft.....              | 4098        |
| ClientAreaTop.....               | 4099        |
| ClientAreaRight.....             | 4100        |
| ClientAreaBottom.....            | 4101        |
| ClientAreaWidth.....             | 4102        |
| ClientAreaHeight.....            | 4103        |
| OnDialogDragStart.....           | 4104        |
| OnDialogDragProcess.....         | 4105        |
| OnDialogDragEnd.....             | 4106        |
| CAppDialog.....                  | 4107        |
| Create.....                      | 4109        |
| Destroy.....                     | 4110        |
| OnEvent.....                     | 4111        |
| Run.....                         | 4112        |
| ChartEvent.....                  | 4113        |
| Minimized.....                   | 4114        |
| IniFileSave.....                 | 4115        |
| IniFileLoad.....                 | 4116        |
| IniFileName.....                 | 4117        |
| IniFileExt.....                  | 4118        |
| CreateCommon.....                | 4119        |
| CreateExpert.....                | 4120        |
| CreateIndicator.....             | 4121        |
| CreateButtonMinMax.....          | 4122        |
| OnClickButtonClose.....          | 4123        |
| OnClickButtonMinMax.....         | 4124        |
| OnAnotherApplicationClose.....   | 4125        |
| Rebound.....                     | 4126        |
| Minimize.....                    | 4127        |
| Maximize.....                    | 4128        |
| CreateInstanceld.....            | 4129        |
| ProgramName.....                 | 4130        |
| SubwinOff.....                   | 4131        |
| <b>27 Übergang von MQL4.....</b> | <b>4132</b> |

## Nachschlagewerk MQL5

MetaQuotes Language 5 (MQL5) ist eingebaute Programmiersprache der Handelsstrategien, ausgearbeitet von der Gesellschaft [MetaQuotes Software Corp.](#), langjährige Erfahrungen in Erzeugung von handelsinformationellen Plattformen. Diese Sprache erlaubt, eigene Programme-Experte zu schreiben (Expert Advisors), die die Verwaltung von Handelsprozessen automatisieren und für die Realisierung eigener Handelsprozesse vollkommen passen. Außerdem kann man in MQL5 eigene technische Indikatoren (Custom Indicators), Scripts und Bibliotheken der Funktionen (Libraries) Erzeugen.

In MQL5 sind viele Funktionen eingeschlossen, die für die Analyse laufender und früherer Preise nötig sind, es gibt Hauptindikatoren und Funktionen für die Verwaltung und Kontrolle der Handelspositionen. Für Schreiben des Programmcodes wird Expertentexteditor MetaEditor 5 benutzt, der verschiedene Konstruktionen der Sprache MQL5 mit Farbe markiert, das erlaubt, sich besser im Text des Expertensystems zu orientieren.

Im griffbereiten Nachschlagewerk gibt es die nach Kategorien aufgeteilten Funktionen, Operationen, reservierte Wörter, andere Sprachkonstruktionen, man kann die Beschreibung jeder benutzten Einheit lernen, die zur Sprache gehört.

Programme die auf MetaQuotes Language 5 geschrieben werden, haben verschiedene Eigenschaften und Funktionen:

- **Ratgeber** – ist ein mechanisches Handelssystem, das mit dem bestimmten Chart verbunden ist. Ratgeber lässt sich ablaufen, wenn das [Ereignis](#), welches der bearbeiten kann, erscheint: Ereignisse der Initialisierung und Deinitialisierung, Ereignis des neuen Ticks, Ereignis des Timers, Ereignis der DOM Veränderung, Ereignis des Charts und Benutzerereignisse.  
Ratgeber kann nicht nur über die potenziellen Deals informieren, sondern auch Deals auf dem Handelskonto automatisch abschließen und sie direkt zum Handelsserver senden. Ratgeber werden im *terminal\_directory\MQL5\Experts* aufbewahren.
- **Benutzerindikator** – technischer Indikator, selbstständig vom Benutzer geschrieben als Zusatz zu den schon integrierten ins Kundenterminal Indikatoren. Benutzerterminals wie auch eingebaute Terminals, können nicht automatisch handeln und sind erst für die Realisierung analytischer Funktionen bestimmt.  
Benutzerindikatoren werden in *Katalog\_des Terminals\MQL5\Indicators* aufbewahren.
- **Script** – Programm für einmalige Ausführung irgendwelcher Handlungen. Zum Unterschied von Experten bearbeiten Scripts keine Ereignisse, außer Starterereignisse (dafür muss im Script die Bearbeiter-Funktion `OnStart` geben). Scripts werden in *terminal\_directory\MQL5\Scripts* aufbewahren.
- **Bibliothek** – Bibliothek von Benutzerfunktionen ist für Speicherung und Verbreitung der oft benutzten Blocks aus Benutzerprogrammen bestimmt. Bibliotheken können sich nicht selbstständig ablaufen lassen. Bibliotheken werden in *terminal\_directory\MQL5\Libraries* aufbewahren.
- **Include Datei** – Ausgangstext der oft benutzten Blocks von Benutzerprogrammen. Solche Dateien können in Ausgangstexte der Experte, Scripts, Benutzerindikatoren und Bibliotheken in der Compilierungsetappe eingeschlossen werden. Benutzen von Include Dateien ist eher bevorzugt, als Benutzen von Bibliotheken wegen der zusätzlichen Nebenkosten beim Aufruf der Bibliotheksfunktionen.  
Include Dateien können sich in demselben Verzeichnis befinden, wie die Ausgangsdatei, in diesem Fall wird die Anweisung [#include](#) mit Doppelanführungszeichen benutzt. Ein anderer Speicherplatz für Include Dateien ist in *terminal\_directory\MQL5\Include*, in diesem Fall wird die Anweisung `#include` mit spitzen Klammern benutzt.

© 2000-2015, [MetaQuotes Software Corp.](#)

## Grundlagen der Sprache

Die Sprache MetaQuotes Language 5 (MQL5) ist eine objektorientierte Programmiersprache der hohen Stufe, die für das Schreiben automatischer Handelsstrategien, technische Benutzerindikatoren für die Analyse verschiedenartiger Finanzmärkte bestimmt ist. Sie erlaubt nicht nur verschiedene Expertensysteme zu schreiben, berufen zur die on-line Arbeit, sondern auch eigene graphische Werkzeuge zu erzeugen, die Handelsentscheidungen zu treffen helfen .

MQL5 zugrunde liegt das Konzept der weit verbreiteten Programmiersprache C++, im Vergleich zu MQL4 sind darin [Auzaelungen](#), [Strukturen](#), [Klassen](#) und [Verarbeitung von Ereignissen](#) . Wegen der Erweiterung der Anzahl eingebauter [Grundtypen](#), ist die Zusammenwirkung ausgeführter Programme auf MQL5 mit anderen Anwendungen mittels dll maximal erleichtert. Syntax der Sprache MQL5 ist der Syntax C++ ähnlich, das erlaubt Programme aus modernen Programmiersprache in diese Sprache leicht zu übertragen.

für Lernen der Sprache MQL5 sind alle Themen nach folgenden Abschnitten gruppiert:

- [Syntax](#)
- [Datentypen](#)
- [Operationen und Ausdrücke](#)
- [Anweisungen](#)
- [Funktionen](#)
- [Variablen](#)
- [Preprozessor](#)
- [objektorientiertes Programmieren](#)



## Syntax

Syntaktisch gesehen ist Programmiersprache der Handelsstrategien MQL5 Programmiersprache C++ sehr ähnlich, außer einiger Möglichkeiten:

- es gibt keine Adressenarithmetik;
- es gibt keinen Identifikator goto;
- anonyme Aufzählung kann nicht vereinbart werden;
- es gibt kein multiple inheritance.

Sehen Sie auch

[Aufzählungen](#), [Strukturen und Klassen](#), [Vererbung](#)

## Kommentare

Mehrzeilige Kommentare fangen mit zwei Symbolen `/*` an und schließen mit zwei Symbolen `*/` ab. Diese Kommentare können nicht eingebettet sein. Einzeilige Kommentare fangen mit zwei Symbolen `//` an, schließen mit dem Symbol der neuen Zeile und können in mehrzeilige Kommentare eingebettet sein. Kommentare sind überall erlaubt, wo Spaces sein können, und jede Anzahl der Spaces zulässt.

### Beispiele:

```
//--- Einzeiliger Kommentar
/* Mehrzeiliges
    // Eingebetteter einzeiliger Kommentar
    Kommentar
*/
```

## Identifikatoren

Identifikatoren werden als Namen für Variablen und Funktionen verwendet. Die Länge des Identifikatoren kann nicht mehr als 63 Zeichen sein.

Zulässige Symbole eines Identifikatoren: Zahlen 0-9, lateinische Gross- und Kleinbuchstaben a-z und A-Z, die als verschiedene Symbole erkannt werden, Unterstreichungszeichen (\_). Das erste Symbol kann keine Zahl sein.

Identifikator muss nicht mit dem [reservierten](#) Wort zusammenfallen.

### Beispiele:

```
NAME1 name1 Total_5 Paper
```

### Sehen Sie auch

[Variablen](#), [Funktionen](#)

## Reservierte Wörter

Die unten aufgezählten Identifikatoren werden als reservierte Wörter fixiert, jedem entspricht die bestimmte Handlung, anders können sie nicht verwendet werden:

### Datentypen

|                 |               |               |
|-----------------|---------------|---------------|
| <u>bool</u>     | <u>enum</u>   | <u>struct</u> |
| <u>char</u>     | <u>float</u>  | <u>uchar</u>  |
| <u>class</u>    | <u>int</u>    | <u>uint</u>   |
| <u>color</u>    | <u>long</u>   | <u>ulong</u>  |
| <u>datetime</u> | <u>short</u>  | <u>ushort</u> |
| <u>double</u>   | <u>string</u> | <u>void</u>   |

### Spezifikatoren des Zuganges

|               |                |                  |
|---------------|----------------|------------------|
| <u>const</u>  | <u>private</u> | <u>protected</u> |
| <u>public</u> | <u>virtual</u> |                  |

### Speicherklassen

|               |              |               |
|---------------|--------------|---------------|
| <u>extern</u> | <u>input</u> | <u>static</u> |
|---------------|--------------|---------------|

### Anweisungen

|                 |             |                 |
|-----------------|-------------|-----------------|
| <u>break</u>    | <u>do</u>   | <u>operator</u> |
| <u>case</u>     | <u>else</u> | <u>return</u>   |
| <u>continue</u> | <u>for</u>  | <u>sizeof</u>   |
| <u>default</u>  | <u>if</u>   | <u>switch</u>   |
| <u>delete</u>   | <u>new</u>  | <u>while</u>    |

### Die übrigen

|              |                |                  |
|--------------|----------------|------------------|
| <u>false</u> | <u>#define</u> | <u>#property</u> |
| <u>this</u>  | <u>#import</u> | <u>template</u>  |

true#includetypename

## Datentypen

Jedes Programm operiert mit Daten. Daten können verschiedener Arten sein abhängig vom Zweck. ZB für Zugang zu Feldelementen werden Felder ganzzahligen Typs verwendet. Preisdaten sind Fließpunktdaten mit doppelter Genauigkeit. Das ist damit verbunden, dass es in der Sprache MQL5 kein Sondertyp für Preisdaten vorausgesehen ist.

Daten verschiedener Typen werden mit verschiedener Geschwindigkeit verarbeitet. Ganzzahlige Daten werden am schnellsten verarbeitet. Für die Verarbeitung der Daten mit doppelter Genauigkeit wird Coprozessor verwendet. Aber wegen der komplizierten inneren Darstellung von Fließpunktdaten werden sie länger als ganzzahlige Daten verarbeitet.

Am längsten werden Zeilendaten verarbeitet. Das ist mit dynamischer Verteilung-Neuverteilung des Operativspeichers verbunden.

Hauptdatentypen:

- ganzzahlige ([char](#), [short](#), [int](#), [long](#), [uchar](#), [ushort](#), [uint](#), [ulong](#))
- logische ([bool](#))
- [Literalwerte](#) (char, uchar)
- Zeilen ([string](#))
- Fließpunktdaten ([double](#), [float](#))
- Farbe ([color](#))
- Datum und Zeit ([datetime](#))
- Aufzählungen ([enum](#))

Zusammengesetzte Datentypen:

- [Strukturen](#);
- [Klassen](#).

In [OOP](#) Termini werden komplizierte Datentypen als abstrakte Datentypen genannt.

Typen color und datetime sind erst für die Darstellung und Parametereingabe sinnvoll, die von aussen festgelegt werden - aus der Tabelle der Experteigenschaften oder aus dem Benutzerindikator (Zubehör "[Inputs](#)"). Datentypen color und datetime werden als Ganzzahle dargestellt. Ganzzahlige Typen mit den Fließpunktdaten werden arithmetische Typen (Zahlentypen) genannt.

In [Ausdrücken](#) wird implizite [Typenreduzierung](#) verwendet, wenn explizite Typenreduzierung nicht angegeben wird.

Sehen Sie auch

[Typenreduzierung](#)

## Ganzzahlige Typen

Ganzzahlige Typen werden in der Sprache MQL5 von elf Arten vertreten. Einige von ihnen können mit den anderen verwendet werden, wenn die Programmlogik es fordert, dabei muss man aber die Regeln der [Typenreduzierung](#) in Aussicht nehmen.

In der Tabelle werden charakteristische Daten jedes Typs angegeben. Außerdem wird in der senkrechten Spalte für jeden Typ der entsprechende Typ in Programmiersprache C++ angegeben.

| Typ                      | Größe in Bytes | minimale Größe             | Maximale Größe                       | Analog in der Sprache C++ |
|--------------------------|----------------|----------------------------|--------------------------------------|---------------------------|
| <a href="#">char</a>     | 1              | -128                       | 127                                  | char                      |
| <a href="#">uchar</a>    | 1              | 0                          | 255                                  | unsigned char, BYTE       |
| <a href="#">bool</a>     | 1              | 0(false)                   | 1(true)                              | bool                      |
| <a href="#">short</a>    | 2              | -32 768                    | 32 767                               | short, wchar_t            |
| <a href="#">ushort</a>   | 2              | 0                          | 65 535                               | unsigned short, WORD      |
| <a href="#">int</a>      | 4              | -2 147 483 648             | 2 147 483 647                        | int                       |
| <a href="#">uint</a>     | 4              | 0                          | 4 294 967 295                        | unsigned int, DWORD       |
| <a href="#">color</a>    | 4              | -1                         | 16 777 215                           | int, COLORREF             |
| <a href="#">long</a>     | 8              | -9 223 372 036 854 775 808 | 9 223 372 036 854 775 807            | __int64                   |
| <a href="#">ulong</a>    | 8              | 0                          | 18 446 744 073 709 551 615           | unsigned __int64          |
| <a href="#">datetime</a> | 8              | 0 (1970.01.01 0:00:00)     | 32 535 244 799 (3000.12.31 23:59:59) | __time64_t                |

Werte der ganzzahligen Typen können als numerische Konstanten, Farbenliterals, Literals Datum-Zeit, [Symbolkonstanten](#) und [Aufzählungen](#).

Sehen Sie auch

[Datenverarbeitung](#), [Konstanten der Zahltypen](#)

## Typen char, short, int und long

### char

Ganzzahliger Typ char nimmt ein Byte (8 Bit) des Speicherplatzes ein und hilft im Dualsystem  $2^8$  Werte=256 auszudrücken. Ganzzahliger Typ char kann sowohl positive, als auch negative Werte. Änderungsbereich ist von -128 bis 127.

### uchar

Ganzzahliger Typ uchar nimmt 1 Byte des Speicherplatzes ein, wie der Typ char, aber zum Unterschied vom Typ char ist uchar ausschliesslich für positive Werte bestimmt. Die minimale Größe beträgt 0, maximale Größe beträgt 255. Der erste Buchstabe im Namen des Types uchar ist die Abkürzung des Wortes unsigned (zeichenlos).

### short

Der ganzzahliger Typ short nimmt 2 Byte (16 Bit) und erlaubt eine Menge der Größen auszudrücken, die  $2$  erhebt in der sechzehnten Potenz ist:  $2^{16}=65\,536$ . Da der Typ short ein Zeichensymbol ist und sowohl positive, als auch negative Größen enthält, ist sein Änderungsbereich von -32 768 bis 32 767.

### ushort

Die zeichenlose Art vom Typ short ist der Typ ushort, dessen Größe auch 2 Byte ist. Minimale Größe ist 0, maximale Größe 65 535.

### int

Ganzzahliger Typ int hat die Größe 4 Byte (32 Bit). Minimale Größe -2 147 483 648, maximale Größe 2 147 483 647.

### uint

Zeichenloser ganzzahliger Typ nimmt 4 Byte des Speicherplatzes ein und erlaubt ganzzahlige Größen von 0 bis 4 294 967 295 auszudrücken.

### long

Ganzzahliger Typ long hat die Größe 8 Byte (64 Bit). Minimale Größe -9 223 372 036 854 775 808, maximale Größe 9 223 372 036 854 775 807.

### ulong

Ganzzahliger Typ ulong nimmt auch 8 Byte ein und erlaubt die Größen von 0 bis 18 446 744 073 709 551 615 zu speichern.

#### Beispiele:

```
char ch=12;
```



```
short sh=-5000;  
int in=2445777;
```

Da zeichenlose ganzzahlige Typen für Speicherung negativer Werte nicht bestimmt sind, kann der Versuch, negative Größe zu erfahren zu unerwarteten Folgen führen. Solch ein Script führt zur Endlosschleife:

```
//--- Endlosschleife  
void OnStart()  
{  
    uchar u_ch;  
  
    for(char ch=-128;ch<128;ch++)  
    {  
        u_ch=ch;  
        Print("ch = ",ch," u_ch = ",u_ch);  
    }  
};
```

Richtig wird auf dieser Weise:

```
//--- richtige Variante  
void OnStart()  
{  
    uchar u_ch;  
  
    for(char ch=-128;ch<=127;ch++)  
    {  
        u_ch=ch;  
        Print("ch = ",ch," u_ch = ",u_ch);  
        if(ch==127) break;  
    }  
};
```

Ergebnis:

```
ch= -128 u_ch= 128  
  
ch= -127 u_ch= 129  
  
ch= -126 u_ch= 130  
  
ch= -125 u_ch= 131  
  
ch= -124 u_ch= 132  
  
ch= -123 u_ch= 133  
  
ch= -122 u_ch= 134  
  
ch= -121 u_ch= 135
```

```
ch= -120 u_ch= 136  
ch= -119 u_ch= 137  
ch= -118 u_ch= 138  
ch= -117 u_ch= 139  
ch= -116 u_ch= 140  
ch= -115 u_ch= 141  
ch= -114 u_ch= 142  
ch= -113 u_ch= 143  
ch= -112 u_ch= 144  
ch= -111 u_ch= 145  
...
```

**Beispiele:**

```
//--- negative Größen können nicht in zeichenlosen Typen gespeichert werden  
uchar  u_ch=-120;  
ushort u_sh=-5000;  
uint   u_in=-401280;
```

Hexadezimal: Zahlen 0-9, Buchstaben a-f oder A-F für Größen 10-15; fangen mit 0x oder 0X an.

**Beispiele:**

```
0x0A, 0x12, 0X12, 0x2f, 0xA3, 0xA3, 0X7C7
```

Sehen Sie auch

[Typenreduzierung](#)

## Symbolkonstanten

Symbole als [Zeilenelemente](#) in MQL5 - sind Indexe in Symbolvorrat Unicode. Sie sind Hexadezimalwerte, die in ganzzahlige Typen umgewandelt werden können und mit denen mit ganzzahligen [Operationen](#) manipuliert werden kann, solche wie Addition und Subtrahieren.

Jedes einzelne Symbol in Anführungszeichen oder hexadezimaler ASCII-Code des Symbols der Art `'\x10'` ist eine Symbolkonstante und hat den Typ `ushort`. ZB Schreibung der Art `'0'` ist ein numerischer Wert 30, der dem Index von Null in der Tabelle der Symbole entspricht.

**Beispiel:**

```
void OnStart()
{
    //--- bestimmen wir Symbolkonstanten
    int symbol_0='0';
    int symbol_9=symbol_0+9; // erhalten wir das Symbol '9'
    //--- geben wir Werte von Konstanten aus
    printf("In dezimaler Form: symbol_0 = %d, symbol_9 = %d",symbol_0,symbol_9);
    printf("In hexadezimaler Form: symbol_0 = 0x%x, symbol_9 = 0x%x",symbol_0,symbol_9);
    //--- fuegen wir Konstanten in eine Zeile hinzu
    string test="";
    StringSetCharacter(test,0,symbol_0);
    StringSetCharacter(test,1,symbol_9);
    //--- so sehen sie in einer Zeile aus
    Print(test);
}
```

Backslash ist ein Kontrollzeichen für Compiler beim Umgang mit Konstantzeilen und Symbolkonstanten im Ausgangstext eines Programms. Einige Symbole, ZB einzelner Anführungsstrich ('), Doppelte Anführungszeichen ("), backslash (\) und Kontrollsymbole können als Symbolkombination dargestellt werden, die mit backslash (\) anfaengt, entsprechend der unten angegebenen Tabelle:

| Symbolname                     | Mnemokode oder Darstellung | Aufzeichnung in MQL5  | Numerischer Wert                    |
|--------------------------------|----------------------------|-----------------------|-------------------------------------|
| neue Zeile<br>(Zeilenvorschub) | LF                         | <code>'\n'</code>     | 10                                  |
| horizontales<br>Tabulieren     | HT                         | <code>'\t'</code>     | 9                                   |
| Vorschub rückgeben             | CR                         | <code>'\r'</code>     | 13                                  |
| backslash                      | <code>\</code>             | <code>'\\'</code>     | 92                                  |
| einzelner<br>Anführungsstrich  | <code>'</code>             | <code>'\''</code>     | 39                                  |
| Doppelanführungszeichen        | <code>"</code>             | <code>'\"'</code>     | 34                                  |
| hexadezimaler Kode             | hhhh                       | <code>'\xhhhh'</code> | von 1 bis 4<br>hexadezimale Zeichen |
| dezimaler Kode                 | d                          | <code>'\d'</code>     | dezimale Zahl von 0<br>bis 65535    |

Wenn das nach slash folgende Symbol sich von de aufgelisteten unterscheidet, ist das Ergebnis unbestimmt.

### Beispiel

```
void OnStart()
{
//--- erklären wir Symbolkonstanten
    int a='A';
    int b='$';
    int c='@';          // Kode 0xA9
    int d='\xAE';       // Symbolkode ®
//--- geben wir Konstante für Abdrucken aus
    Print(a,b,c,d);
//--- fuegen wir ein Symbol in die Zeile zu
    string test="";
    StringSetCharacter(test,0,a);
    Print(test);
//--- Ersetzen wir Symbol in einer Zeile
    StringSetCharacter(test,0,b);
    Print(test);
//--- Ersetzen wir Symbol in einer Zeile
    StringSetCharacter(test,0,c);
    Print(test);
//--- Ersetzen wir Symbol in einer Zeile
    StringSetCharacter(test,0,d);
    Print(test);
//--- stellen wir Symbole als eine Zahl dar
    int a1=65;
    int b1=36;
    int c1=169;
    int d1=174;
//--- setzen wir Symbol in die Zeile zu
    StringSetCharacter(test,1,a1);
    Print(test);
//--- ersetzen wir Symbol in einer Zeile
    StringSetCharacter(test,1,b1);
    Print(test);
//--- ersetzen wir Symbol in einer Zeile
    StringSetCharacter(test,1,c1);
    Print(test);
//--- ersetzen wir Symbol in einer Zeile
    StringSetCharacter(test,1,d1);
    Print(test);
}
```

Wir es schon oben angegeben wurde, ist der Wert von einer Symbolkonstante (oder Variable) Index in der Symboltabelle darstellt, da ein Index eine Ganzzahl ist, ist es zulaessig es verschieden zu schreiben.

```

void OnStart()
{
//---
    int a=0xAE;      // Symbolkode ® entspricht dem Literal '\xAE'
    int b=0x24;      // Symbolkode $ entspricht dem Literal '\x24'
    int c=0xA9;      // Symbolkode © entspricht dem Literal '\xA9'
    int d=0x263A;    // Symbolkode O entspricht dem Literal '\x263A'
//--- geben wir Werte aus
    Print(a,b,c,d);
//--- fuegen wir Symbol in die Zeile hinzu
    string test="";
    StringSetCharacter(test,0,a);
    Print(test);
//--- ersetzen wir Symbol in einer Zeile
    StringSetCharacter(test,0,b);
    Print(test);
//--- ersetzen wir Symbol in einer Zeile
    StringSetCharacter(test,0,c);
    Print(test);
//--- ersetzen wir Symbol in einer Zeile
    StringSetCharacter(test,0,d);
    Print(test);
//--- Kodes der Farben
    int a1=0x2660;
    int b1=0x2661;
    int c1=0x2662;
    int d1=0x2663;
//---fuegen wir Pikssymbol hinzu
    StringSetCharacter(test,1,a1);
    Print(test);
//--- fuegen wir Herzsymbol hinzu
    StringSetCharacter(test,2,b1);
    Print(test);
//--- fuegen wir Karosymbol hinzu
    StringSetCharacter(test,3,c1);
    Print(test);
//--- fuegen wir Treffsymbol hinzu
    StringSetCharacter(test,4,d1);
    Print(test);
//--- Beispiel der Symbolliterale in der Zeile
    test="Dame\x2660As\x2662";
    printf("%s",test);
}

```

Interne Darstellung des Symbolliterals ist Typ [ushort](#). Symbolkonstanten können die Werte von 0 bis 65535 annehmen.

#### Sehen Sie auch

[StringSetCharacter\(\)](#), [StringGetCharacter\(\)](#), [ShortToString\(\)](#), [ShortArrayToString\(\)](#),  
[StringToShortArray\(\)](#)

## Typ datetime

Typ **datetime** ist für Datum- und Zeitspeicherung als Sekundenanzahl bestimmt, die seit 01 Januar 1970 vergangen ist. Es nimmt 8 Byte des Speicherplatzes ein.

Daten- und Zeitkonstanten können als Literalzeile dargestellt werden, die aus 6 Teilen besteht. Diese Teile repräsentieren die Zahlgrösse des Jahres, Monats, der Zahl (oder des Monatstages, des Monats, des Jahres), der Stunde, der Minute und der Sekunde. Konstante wird in einzelnen Anführungsstrich gesetzt und faengt mit Symbol D an.

Änderungsbereich vom 1 Januar 1970 bis 31 Dezember 3000. Fallen kann entweder das Datum (Jahr, Monat, Monatstag) oder die Zeit (Stunden, Minuten, Sekunden), oder alles zusammengenommen.

In der wörtlichen Einstellung des Datums ist wünschenswert, den Jahr, Monat und Tag anzugeben, sonst zeigt der Compiler [Warnungen](#) über unvollständigen wörtlichen Eintrag.

### Bespiele:

```
datetime NY=D'2015.01.01 00:00'; // Zeit der Anfang des Jahres 2015
datetime d1=D'1980.07.19 12:30:27'; // Jahr Monat Tag Stunden Minuten Sekunden
datetime d2=D'19.07.1980 12:30:27'; // äquivalent zu D'1980.07.19 12:30:27';
datetime d3=D'19.07.1980 12'; // äquivalent zu D'1980.07.19 12:00:00'
datetime d4=D'01.01.2004'; // äquivalent zu D'01.01.2004 00:00:00'
datetime compilation_date=__DATE__; // Datum der Kompilierung
datetime compilation_date_time=__DATETIME__; // Datum und Zeit der Kompilierung
datetime compilation_time=__DATETIME__-__DATE__; // Zeit der Kompilierung
//--- Beispiele, in denen Compiler/Warnungen zurückgegeben werden
datetime warning1=D'12:30:27'; // äquivalent zu D'[Datum der Kompilierung] 12:30:27'
datetime warning2=D''; // äquivalent zu __DATETIME__
```

### Sehen Sie auch

[Datenstruktur](#), [Datum und Zeit](#), [TimeToString](#), [StringToTime](#)

## Typ color

Typ **color** ist für die Speicherung farbenbezogener Information bestimmt und nimmt 4 Bytes des Speicherplatzes ein. Das erste Byte wird nicht berücksichtigt, die anderen 3 Bytes enthalten RGB-Komponenten.

Farbkonstanten können durch drei verschiedene Arten vertreten werden: literal, ganzzahlig oder mit Hilfe des Namens (nur für die genannten [Web-Farben](#)).

Literaldarstellung besteht aus drei Teilen. Sie stellen die Zahlgrößen der Intensität von drei Hauptfarbarten: rot (red), grün (green), blau (blue). Konstante beginnt mit Symbol c und wird in einzelne Anführungszeichen gesetzt. Zahlgrößen der Intensität von Farbkomponenten sind von 0 bis 255.

Ganzzahlige Darstellung wird als Hexadezimalzahl oder als Dezimalzahl eingetragen. Hexadezimalzahl ist der Art 0x00BBGGRR, wo RR - Intensitätsgrösse der roten Farbkomponente, GG - der grünen, BB - der blauen. Dezimalkonstanten haben keine direkte Darstellung in RGB. Sie bilden Dezimalgrösse der ganzzahligen Darstellung.

Benannte Farben spiegeln den sogenannten Bestand von [Web-Farben](#) wieder.

### Beispiele:

```
//--- Literalwerte
C'128,128,128'    // grau
C'0x00,0x00,0xFF' // blau
//Farbbezeichnungen
clrRed           // rot
clrYellow        // gelb
clrBlack         // schwarz
//--- ganzzahlige Darstellungen
0xFFFFFFFF       // weiss
16777215         // weiss
0x008000         // grün
32768            // grün
```

### Sehen Sie auch

[Web-Farben](#), [ColorToString](#), [StringToColor](#), [Typenreduzierung](#)

## Typ bool

Typ **bool** ist für die Speicherung logischer Werte **true** (richtig) oder **false** (falsch) bestimmt, deren Zahldarstellung 1 oder 0 ist.

### Beispiele:

```
bool a = true;
bool b = false;
bool c = 1;
```

Interne Darstellung ist die Ganzzahl mit der Größe 1 Byte. Es muss hervorgehoben werden, dass es in logischen Ausdrücken zulaessig ist, anstatt des Typs bool andere ganzzahlige Typen oder Realtypen oder Ausdrücke dieser Typen zu verwenden, Compiler wird kein Fehler zeigen. In diesem wird der Wert 0 als false interpretiert, alle andere Werte als true.

### Beispiele:

```
int i=5;
double d=-2.5;
if(i) Print("i = ",i," und Wert true hat");
else Print("i = ",i," und Wert false hat");

if(d) Print("d = ",d," und Wert true hat");
else Print("d = ",d," und Wert false hat");

i=0;
if(i) Print("i = ",i," und Wert true hat");
else Print("i = ",i," und Wert false hat");

d=0.0;
if(d) Print("d=",d,"und Wert true hat");
else Print("d=",d,"und Wert false hat");

//--- Ergebnisse der Ausführung
//   i= 5 und Wert true hat
//   d= -2.5 und Wert true hat
//   i= 0 und Wert false hat
//   d= 0 und Wert false hat
```

### Sehen Sie auch

[Logische Operationen](#), [Prioritaeten und Operationsordnung](#)



## Aufzählungen

Daten des Typs Aufzählungen `enum` gehören zur einer begrenzten Datenmenge. Definition des aufzählbaren Typs:

```
enum Name des aufzählbaren Typs
{
    Liste von Bedeutungen
};
```

Liste von Bedeutungen stellt Konstantenliste dar, die durch Kommas voneinander getrennt sind.

**Beispiel:**

```
enum months // Aufzählung der benannten Konstanten
{
    January,
    February,
    March,
    April,
    May,
    June,
    July,
    August,
    September,
    October,
    November,
    December
};
```

Nach der Vereinbarung von Aufzählung erscheint ein neuer ganzzahliger 4-Byte Datentyp. Vereinbarung des neuen Datentyps ermöglicht dem Compiler Typen der übertragenen Parameter strikt zu kontrollieren, weil die Aufzählung neue benannten Konstanten einführt. In dem angeführten Beispiel hat die benannte Konstante die Größe 0, February hat die Größe 1, December hat die Größe 11.

Regel: Wenn einer benannten Konstante - dem Aufzählungsglied keinen konkreten Wert zugeordnet wird, wird ihr Wert automatisch erzeugt. Wenn es das Erste Aufzählungsglied ist, wird der Wert 0 zugeordnet. Für alle folgenden Glieder werden die Werter aufgrund des Wertes vom vorangehenden Gliedes aufgezählt und eine Eins hinzugefügt.

**Beispiel:**

```
enum intervals // Aufzählung der benannten Konstanten
{
    month=1, // Interval aus einem Monat
    two_months, // zwei Monate
    quarter, // drei Monate - Quartal
    halfyear=6, // Halbjahr
    year=12, // Jahr - 12 Monate
};
```

## Bemerkungen

- Zum Unterschied von C++, ist die Größe der inneren Darstellung des aufzählbaren Typs in MQL5 immer 4 Byte. D.h. [sizeof](#)(months) gibt den Wert 4 zurück.
- Zum Unterschied von C++, kann in MQL5 keine anonyme Aufzählung erklärt werden. D.h. nach Schlüsselwort enum kann immer unikaler Name angegeben werden.

### Sehen Sie auch

[Typenreduzierung](#)

## Realtypen (double, float)

Realtypen (Fließpunkttypen) stellen die Werte mit Bruchanteil dar. In der Sprache MQL5 gibt es zwei Typen für Fließpunktzahlen. Darstellungsart der Realzahlen im Computerspeicher ist vom Standard IEEE 754 bestimmt und hängt nicht von Plattformen, Betriebssystemen und Programmiersprachen ab.

| Typ    | Größe in Bytes | minimale positive Größe | Maximale Größe          | Genauigkeit der Darstellung | Analog in C++ |
|--------|----------------|-------------------------|-------------------------|-----------------------------|---------------|
| float  | 4              | 1.175494351e-38         | 3.402823466e+38         | 7 bedeutsame Ziffern        | float         |
| double | 8              | 2.2250738585072014e-308 | 1.7976931348623158e+308 | 15 bedeutsame Ziffern       | double        |

Name **double** bedeutet, dass die Genauigkeit dieser Zahlen um das Doppelte die Genauigkeit der Zahlen der Art **float** übertrifft. In meistens Faellen ist die Art **double** am bequemsten. Die begrenzte Genauigkeit der Zahle **float** ist oft nicht ausreichend. Der Grund, warum **float** noch verwendet wird, - Speicheroekonomie bei der Bewahrung (das ist sehr wichtig für grosse Felder der Realzahlen).

Konstanten mit Fließpunkt bestehen aus dem ganzzahligen Teil, Punkt (.) und Bruchanteil. Ganzzahliger Teil und Bruchanteil stellen Folgen von Dezimalzahlen dar.

### Beispiele:

```
double a=12.111;  
double b=-956.1007;  
float c =0.0001;  
float d =16;
```

Es gibt ein wissenschaftliches Schreibverfahren der Realkonstanten. Oft ist dieses Schreibverfahren kompakter, als das traditionelle.

### Beispiel:

[illegible]

Man muss im Kopf behalten, dass Realzahlen im Bionäersystem mit begrenzter Genauigkeit gespeichert werden, während das allgemeinguelgige System Dezimalsystem ist. Darum können viele

Zahlen, die genau im Dezimalsystem geschrieben werden, können im Dualsystem nur als Infinitesimalbruch geschrieben werden.

ZB. werden die Zahlen 0.3 und 0.7 im Computer als Infinitesimalbrueche vertreten, während die Zahl 0, 25 wird genau gespeichert, da es die Potenz der zwei ist.

In diesem Zusammenhang ist es nicht empfehlenswert, zwei Realzahlen miteinander zu vergleichen, weil dieser Vergleich nicht korrekt ist.

#### Beispiel:

```
void OnStart()
{
//---
double three=3.0;
double x,y,z;
x=1/three;
y=4/three;
z=5/three;
if(x+y==z) Print("1/3 + 4/3 == 5/3");
else Print("1/3 + 4/3 != 5/3");
// Ergebnis: 1/3 + 4/3 != 5/3
;}
```

Wenn es aber notwendig ist, zwei Realzahlen miteinander zu vergleichen, kann es mit zwei Verfahren gemacht werden. Das erste Verfahren besteht im Vergleich des Differenzbetrages der zwei Zahlen mit irgendwelcher kleinen Größe, die die Genauigkeit des Vergleiches festlegt.

#### Beispiel:

```
bool EqualDoubles(double d1,double d2,double epsilon)
{
if(epsilon<0) epsilon=-epsilon;
//---
if(d1-d2>epsilon) return false;
if(d1-d2<-epsilon) return false;
//---
return true;
; }

void OnStart()
{
double d_val=0.7;
float f_val=0.7;
if(EqualDoubles(d_val,f_val,0.0000000000000001)) Print(d_val,"equals",f_val);
else Print("Different: d_val=",DoubleToString(d_val,16),
" f_val=",DoubleToString(f_val,16));
// Ergebnis: Different: d_val= 0.7000000000000000 f_val= 0.6999999880790710
; }
```

Es muss bemerkt werden, dass der Wert von epsilon im angeführten Beispiel nicht kleiner als der der internen Konstante DBL\_EPSILON sein kann. Der Wert dieser Konstante ist 2.2204460492503131e-016.

Für den Typ float ist die entsprechende Konstante FLT\_EPSILON = 1.192092896e-07. Der Sinn dieser Werter besteht darin, dass es der minimale Wert ist, der der Bedingung entspricht  $1.0 + \text{DBL\_EPSILON} \neq 1.0$  (für die Zahlen float  $1.0 + \text{FLT\_EPSILON} \neq 1.0$ ).

Das zweite Verfahren setzt den Vergleich normalisierter Differenz von zwei Realzahlen mit dem Nullwert. Es ist sinnlos, Differenz normalisierter Zahlen mit Null zu vergleichen, denn im Ergebnis jeder mathematischen Operation mit normalisierter Zahlen wird das Ergebnis nicht normalisiert.

#### Beispiel:

```
bool CompareDoubles(double number1, double number2)
{
    if(NormalizeDouble(number1-number2, 8) == 0) return(true);
    else return(false);
};
void OnStart()
{
    double d_val=0.3;
    float f_val=0.3;
    if(CompareDoubles(d_val, f_val)) Print(d_val, "equals", f_val);
    else Print("Different: d_val=", DoubleToString(d_val, 16),
        " f_val=", DoubleToString(f_val, 16));
    // Ergebnis: Different: d_val= 0.3000000000000000    f_val= 0.3000000119209290
};
```

Im Ergebnis einiger Operationen des mathematischen Coprozessors kann eine ungueltige Realzahl entstehen, die nicht in mathematischen Operationen und Vergleichsoperationen verwendet werden kann, denn das Ergebnis der Operationen mit ungueltigen Realzahlen ist nicht bestimmt. ZB beim Versuch [arc sin](#) 2 auszurechnen, bekommt man das Ergebnis minus Infinitaet.

#### Beispiel:

```
double abnormal = MathArcsin(2.0);
Print("MathArcsin(2.0) =", abnormal);
// Ergebnis: MathArcsin(2.0) = -1.#IND
```

Ausser minus Infinitaet gibt es auch plus Infinitaet und NaN (keine Zahl). Um festzustellen, das diese Zahl ungueltig ist, kann man die funktion [MathIsValidNumber\(\)](#) benutzen Nach Standard IEEE haben Sie spezielle Maschienenendarstellung. ZB plus Infinitaet für den Typ double hat Bitdarstellung 0x7FF0 0000 0000 0000.

#### Beispiele:

```
struct str1
{
    double d;
};
struct str2
{
    long l;
};
```

```
//--- fangen wir an
    str1 s1;
    str2 s2;
//---
    s1.d=MathArcsin(2.0);          // wir bekommen ungueltige Zahl -1.#IND
    s2=s1;
    printf("1.   %f %I64X",s1.d,s2.l);
//---
    s2.l=0xFFFF000000000000;      // ungueltige Zahl -1.#QNAN
    s1=s2;
    printf("2.   %f %I64X",s1.d,s2.l);
//---
    s2.l=0x7FF7000000000000;      // maximale Nichtzahl SNaN
    s1=s2;
    printf("3.   %f %I64X",s1.d,s2.l);
//---
    s2.l=0x7FF8000000000000;      // minimale Nic QNhtzahlaN
    s1=s2;
    printf("4.   %f %I64X",s1.d,s2.l);
//---
    s2.l=0x7FFF000000000000;      // maximale Nichtzahl QNaN
    s1=s2;
    printf("5.   %f %I64X",s1.d,s2.l);
//---
    s2.l=0x7FFF000000000000;      // plus Infinitum 1.#INF und minimale Nichtzahl SNaN
    s1=s2;
    printf("6.   %f %I64X",s1.d,s2.l);
//---
    s2.l=0xFFFF000000000000;      // minus Infinitum -1.#INF
    s1=s2;
    printf("7.   %f %I64X",s1.d,s2.l);
//---
    s2.l=0x8000000000000000;      // negative Null -0.0
    s1=s2;
    printf("8.   %f %I64X",s1.d,s2.l);
//---
    s2.l=0x3FE0000000000000;      // 0.5
    s1=s2;
    printf("9.   %f %I64X",s1.d,s2.l);
//---
    s2.l=0x3FF0000000000000;      // 1.0
    s1=s2;
    printf("10.  %f %I64X",s1.d,s2.l);
//---
    s2.l=0x7FEFFFFFFFFFFFFFFF;    // minimale normalisierte Zahl (MAX_DBL)
    s1=s2;
    printf("11.  %.16e %I64X",s1.d,s2.l);
//---
    s2.l=0x0010000000000000;      // minimale positive normalisierte (MIN_DBL)
```

```
s1=s2;
printf("12.  %.16e %.16I64X",s1.d,s2.l);
//---
s1.d=0.7;           // zeigen wir, dass die Zahl 0.7 - Infinitesimalbruch
s2=s1;
printf("13.  %.16e %.16I64X",s1.d,s2.l);
/*
1.  -1.#IND00 FFF8000000000000
2.  -1.#QNAN0 FFFF000000000000
3.   1.#SNAN0 7FF7000000000000
4.   1.#QNAN0 7FF8000000000000
5.   1.#QNAN0 7FFF000000000000
6.   1.#INF00 7FF0000000000000
7.  -1.#INF00 FFF0000000000000
8.  -0.000000 8000000000000000
9.   0.500000 3FE0000000000000
10.  1.000000 3FF0000000000000
11.  1.7976931348623157e+308 7FFFFFFFFFFFFFFF
12.  2.2250738585072014e-308 0010000000000000
13.  6.9999999999999996e-001 3FE6666666666666
*/
```

#### Sehen Sie auch

[DoubleToString](#), [NormalizeDouble](#), [Konstanten der numerischen Datentypen](#)

## Typ string

Typ string ist für die Speicherung von Textzeilen bestimmt. Textzeile ist eine Symbolenfolge in Unicode Format mit Null am Ende. Zeilenkonstante ist eine Symbolfolge in Unicode mit doppelten Anführungszeichen: "Das ist eine Zeilenkonstante"

Wenn es notwendig ist, in die Zeile ein Doppelanführungszeichen einzugeben, muss man vor diesem Zeichen einen nach links weisender Schrägstrich (\) stellen. In die Zeile können verschiedene [Sondersymbolkonstanten](#) eingegeben werden, vor denen ein nach links weisender Schrägstrich steht (\).

### Beispiele:

```
string svar="This is a character string";
string svar2=StringSubstr(svar,0,4);
Print("Copyright Symbol\t\x00A9");
FileWrite(handle,"Diese Zeile hat das CR-Zeichen \n");
string T5path="C:\\Program Files\\MetaTrader 5";
```

Um den Quellcode lesbar zu machen, können Sie lange konstante Strings in Teile ohne Zusatz Operationen aufteilen. Beim Kompilierung werden diese Teile automatisch in einem langen String kombiniert werden:

```
//--- Ein langer konstanter Sting deklarieren
string HTML_head="<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML 1.0 Transitional//EN\"
    \"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd\">\n"
    "<html xmlns=\"http://www.w3.org/1999/xhtml\">\n"
    "<head>\n"
    "<meta http-equiv=\"Content-Type\" content=\"text/html; charset=ut
    <title>Trade Operations Report</title>\n"
    "</head>";
//--- Den konstanten String im Journal anzeigen
Print(HTML_head);
; }
```

### Sehen Sie auch

[Datenverarbeitung](#), [Zeilenfunktionen](#), [FileOpen](#), [FileReadString](#), [FileWriteString](#)



## Strukturen und Klassen

### Strukturen

Struktur ist Bestand von Elementen eines willkuerlichen Typs (Ausser Typ [void](#)). So gruppiert Struktur logisch verbundene Daten verschiedener Typen.

### Strukturvereinbarung

Strukturdatentyp ist durch die folgende Beschreibung bestimmt:

```
struct Strukturname
{
    Elementenbeschreibung
};
```

Strukturname kann nicht als Identifikator verwendet werden (Name der Variable oder Funktion). Es muss in Vormerkung behalten, dass Strukturelemente in MQL5 folgen unmittelbar nacheinander ohne Ausrichtung. In der Sprache C++ wird diese Festlegung an den Compiler mit der Anweisung gerichtet.

```
#pragma pack(1)
```

Wenn es notwendig ist, eine andere Ausrichtung in Struktur durchzuführen, muss man "Fuellzeichen" notwendiger Größen verwenden.

### Beispiel:

```
struct trade_settings
{
    uchar slippage;    // Wert der zulaessigen Glaettung - Größe 1 Byte
    char  reserved1;   // 1 Byte Leerplatzes
    short reserved2;   // 2 Byte Leerplatzes
    int   reserved4;   // noch 4 Byte Leerplatzes. Richtung der Grenze 8 Byte garanti
    double take;       // Preiswert der Gewinnfixierung
    double stop;       // Preiswert des Schutzstops
};
```

Solche Beschreibung ausgerichteter Strukturen ist nur für die Übergabe in importierte dll-Funktionen notwendig.

Achtung: dieses Beispiel zeigt Falsch projektierte Daten. Es wäre besser vor allem Daten take und stop mit dem größeren Wert zu vereinbaren der Art [double](#) zu vereinbaren, und dann das Glied slippage der Art uchar zu vereinbaren. In diesem Fall wird immer die interne Darstellung von Daten dieselbe, unabhängig vom Wert angegeben in `#pragma pack()`.

Wenn die Struktur Variablen der Art [string](#) und/oder [Objekt des dynamischen Feldes](#) enthält, ordnet der Compiler für diese Struktur einen impliziten Konstruktor zu, in dem alle Strukturglieder der Art htige für Objekt des dynamischen Feldes vorkommt. Variablen einfacher Strukturen und ihre Felder können als Parameter in die aus DLL [importierten](#) Funktionen.

### Einfache Strukturen

Strukturen ohne Zeilen und Objekte des dynamischer Felder werden einfache Strukturen genannt; Variablen solcher Strukturen können [frei in einander kopiert](#) werden, auch wenn es sich um verschiedene Strukturen handelt.

## Zugang zu Strukturgliedern

Strukturname ist ein neuer Datentyp und ermöglicht Variablen dieses Typs zu vereinbaren. Struktur kann nur einmal im Rahmen dieses Projekts vereinbart werden. Zugang zu Strukturgliedern ist durch die [Operation Punkt](#) (.) durchgeführt.

**Beispiel:**

```
struct trade_settings
{
    double take;           // Preiswert von Gewinnmitnahme
    double stop;           // Preiswert von Schutzstop
    uchar slippage;        // Wert der zulaessigen Glaettung
};
//--- Variable des Typs trade_settings ist erzeugt und initialisiert
trade_settings my_set={0.0,0.0,5};
if (input_TP>0) my_set.take=input_TP;
```

## Klassen

Klassen haben eine Reihe Unterschiede von den Strukturen:

- in der Vereinbarung wird das Schlüsselwort `class` verwendet;
- alle Glieder der Klasse haben einen impliziten Spezifikator `private`, wenn nicht anders angegeben wird. Glieder-Daten der Struktur haben impliziten Zugangstyp `public`, wenn nicht anders angegeben wird;
- Klassenobjekte haben immer die Tabelle [virtueller Funktionen](#), auch wenn es keine einzige virtuelle Funktion in der Klasse erklärt wird. Strukturen können keine virtuelle Funktionen haben;
- Operator [new](#) kann für Klassenobjekte verwendet werden, für Strukturen kann dieser Operator nicht verwendet werden;
- Klassen können nur von Klassen [heritiert werden](#), Strukturen können nur von Strukturen heritiert werden.

Klassen können offenbaren Konstruktor und Destruktor haben. Wenn der Konstruktor bestimmt wird, ist Initialisierung der Variable des Typs Struktur oder Klasse durch Initialisierungsfolge unmöglich.

**Beispiel:**

```
struct trade_settings
{
    double take;           // Preiswerte von Gewinnmitnahme
    double stop;           // Preiswerte von Schutzstop
    uchar slippage;        // Wert der zulaessigen Glaettung
    //--- Konstruktor
    trade_settings() { take=0.0; stop=0.0; slippage=5; }
    //--- Destruktor
```

```

~trade_settings() { Print("Das ist Ende"); }

};

//--- Compiler generiert eine Fehlermeldung, dass Initialisierung unmöglich ist
trade_settings my_set={0.0,0.0,5};

```

## Konstruktoren und Destruktoren

Konstruktor ist eine spezielle Funktion, die automatisch bei Erstellung von Struktur- oder Klassenobjekt aufgerufen. Er wird normalerweise und für [Initialisierung](#) von Klassenglieder verwendet. Weiter werden wir nur über die Klassen reden, während das gleiche gilt für Strukturen, soweit nicht anders angegeben. Der Name eines Konstruktors muss mit dem Klassennamen gleich sein. Der Konstruktor hat keinen Rückgabotyp (Sie können den Typ [void](#) angeben).

Definierte Klasseglieder - [Strings](#), [dynamische Arrays](#) und Objekte, die Initialisierung erfordern, werden in jedem Fall initialisiert werden, unabhängig davon, ob es ein Konstruktor gibt.

Jede Klasse kann mehrere Konstruktoren mit unterschiedlichen Anzahl von Parametern und Initialisierungsliste haben. Der Konstruktor, der Angabe von Parametern erfordert, heißt parametrischer Konstruktor.

Ein Konstruktor ohne Parameter ist ein **Default-Konstruktor**. Wenn keine Konstruktoren in einer Klasse deklariert sind, erstellt der Compiler einen Default-Konstruktor während der Kompilierung.

```

//+-----+
//| Klasse für Arbeiten mit Datum |
//+-----+
class MyDateClass
{
private:
    int          m_year;          // Jahr
    int          m_month;         // Monat
    int          m_day;           // Tag des Monats
    int          m_hour;          // Stunde an einem Tag
    int          m_minute;        // Minuten
    int          m_second;        // Sekunden
public:
    //--- Default-Konstruktor
    MyDateClass(void);

    //--- parametrischer Konstruktor
    MyDateClass(int h,int m,int s);
};

```

Sie können ein Konstruktor in der Klassebeschreibung deklarieren und dann ihren Körper definieren. Zum Beispiel zwei Konstruktoren der Klasse MyDateClass können auf folgende Weise definiert werden:

```

//+-----+
//| Default-Konstruktor |
//+-----+
MyDateClass::MyDateClass(void)
{

```

```
//---
MqlDateTime mdt;
datetime t=TimeCurrent(mdt);
m_year=mdt.year;
m_month=mdt.mon;
m_day=mdt.day;
m_hour=mdt.hour;
m_minute=mdt.min;
m_second=mdt.sec;
Print(__FUNCTION__);
;}

//+-----+
//| Parametrischer Konstruktor |
//+-----+
MyDateClass::MyDateClass(int h,int m,int s)
{
    MqlDateTime mdt;
    datetime t=TimeCurrent(mdt);
    m_year=mdt.year;
    m_month=mdt.mon;
    m_day=mdt.day;
    m_hour=h;
    m_minute=m;
    m_second=s;
    Print(__FUNCTION__);
;}
```

Im Default-Konstruktor werden alle Klassenglieder mit Hilfe der Funktion TimeCurrent() gefüllt. Im parametrischen Konstruktor werden nur Stunde gefüllt. Andere Glieder der Klasse (m\_year, m\_month und m\_day) werden automatisch mit dem aktuellen Datum automatisch initialisiert.

Der Default-Konstruktor hat eine besondere Bedeutung bei der Initialisierung eines Arrays von Objekten seiner Klasse. Konstruktor, denen alle Parameter Standardwerten haben, ist kein Default-Konstruktor. Wir zeigen dies am Beispiel:

```
//+-----+
//| Klasse mit einem Default-Konstruktor |
//+-----+
class CFoo
{
    datetime      m_call_time;      // Zeit des letzten Aufrufs des Objektes
public:
    //--- Konstruktor mit einem Parameter, der einen Standardwert hat, ist kein Default
        CFoo(const datetime t=0){m_call_time=t;};
    //--- Copy-Konstruktor
        CFoo(const CFoo &foo){m_call_time=foo.m_call_time;};

    string ToString(){return(TimeToString(m_call_time,TIME_DATE|TIME_SECONDS));};
};

//+-----+
```

```

//| Script program start function |
//+-----+
void OnStart()
{
// CFoo foo; // diese Option kann nicht verwendet werden - Default-Konstruktor ist nicht definiert
//--- Gültige Optionen für Erstellung des Objekts CFoo
CFoo foo1(TimeCurrent()); // Ein expliziter Aufruf des parametrischen Konstruktors
CFoo foo2(); // Ein expliziter Aufruf des parametrischen Konstruktors
CFoo foo3=D'2009.09.09'; // Ein impliziter Aufruf des parametrischen Konstruktors
CFoo foo4(foo1); // Ein expliziter Aufruf des Copy-Konstruktors
CFoo foo41=foo1; // Ein impliziter Aufruf des Copy-Konstruktors
CFoo foo5; // Ein expliziter Aufruf des Default-Konstruktors (wenn nicht definiert, wird ein parametrischer Konstruktor mit Standardparametern verwendet)

//--- Gültige Optionen für Erlangung von Zeigern CFoo
CFoo *pfoo6=new CFoo(); // dynamische Erstellung des Objekts und Erhaltung des Objekts
CFoo *pfoo7=new CFoo(TimeCurrent()); // Eine andere Variante der dynamischen Erstellung
CFoo *pfoo8=GetPointer(foo1); // pfoo8 zeigt nun auf Objekt foo1
CFoo *pfoo9=pfoo7; // pfoo9 und pfoo7 zeigen auf das gleiche Objekt
// CFoo foo_array[3]; // diese Option kann nicht verwendet werden - Default-Konstruktor ist nicht definiert
//--- Ableiten Werte m_call_time
Print("foo1.m_call_time=",foo1.ToString());
Print("foo2.m_call_time=",foo2.ToString());
Print("foo3.m_call_time=",foo3.ToString());
Print("foo4.m_call_time=",foo4.ToString());
Print("foo5.m_call_time=",foo5.ToString());
Print("pfoo6.m_call_time=",pfoo6.ToString());
Print("pfoo7.m_call_time=",pfoo7.ToString());
Print("pfoo8.m_call_time=",pfoo8.ToString());
Print("pfoo9.m_call_time=",pfoo9.ToString());
//--- Dynamisch erstellte Objekte löschen
delete pfoo6;
delete pfoo7;
//delete pfoo8; // Es ist nicht notwendig pfoo8 explizit zu löschen, da er auf das Objekt foo1 zeigt
//delete pfoo9; // Es ist nicht notwendig pfoo9 explizit zu löschen, da er auf das Objekt pfoo7 zeigt
};

```

Wenn Sie kommentieren Zeile in diesem Beispiel

```
//CFoo foo_array[3]; // diese Option kann nicht verwendet sein - kein Default-Konstruktor
```

oder

```
//CFoo foo_dyn_array[]; // diese Option kann nicht verwendet sein - kein Default-Konstruktor
```

dann wird der Compiler Fehler "default constructor is not defined" zurückgeben.

Wenn die Klasse hat einen Konstruktor, der vom Benutzer definiert ist, wird der Default-Konstruktor vom Compiler nicht generiert. Dies bedeutet, dass, wenn ein parametrischer Konstruktor in einer Klasse deklariert war, aber kein Default-Konstruktor war deklariert, kann man die Arrays von Objekten dieser Klasse nicht deklarieren. Auf diesem Script meldet der Compiler einen Fehler:

```
//+-----+
//| Eine Klasse ohne Default-Konstruktor |
//+-----+
class CFoo
{
    string          m_name;
public:
    CFoo(string name) { m_name=name;}

};

//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    //-- Fehler "default constructor is not defined" ist während Kompilierung zurückgekehrt
    CFoo badFoo[5];
};
```

In diesem Beispiel hat die Klasse CFoo einen deklarierten parametrischen Konstruktor - in solchen Fällen erstellt der Compiler keinen Default-Konstruktor während der Kompilierung. Zur gleichen Zeit, wenn Sie ein Array von Objekten deklarieren, ist es vorausgesetzt, dass alle müssen [automatisch erstellt und initialisiert werden](#). Beim automatischen Initialisierung eines Objekts, ein Default-Konstruktor muss aufgerufen werden, aber da der Default-Konstruktor nicht explizit deklariert war, und nicht automatisch durch den Compiler erstellt war, Erstellung eines solchen Objekts ist unmöglich. Aus diesem Grund generiert der Compiler einen Fehler bei der Kompilierung.

Es gibt eine spezielle Syntax, um das Objekt mit dem Konstruktor zu initialisieren. Konstruktor-Initializers (spezielle Konstruktionen für Initialisierung) für die Glieder einer Struktur oder Klasse können in der Initialisierungsliste angegeben werden.

Initialisierungsliste ist eine Liste von durch Kommas getrennten Initialisierern, der durch einen Doppelpunkt nach der [Parameterliste](#) des Konstruktors und vor dem [Körper](#) kommt (kommt vor der öffnenden Klammer). Es gibt mehrere Anforderungen:

- Initialisierungslisten können nur in [Konstruktoren](#) verwendet werden.
- [Mitglieder von Eltern](#) können nicht in der Initialisierungsliste initialisiert werden
- Nach der Initialisierungsliste muss [Definition](#) (Implementierung) der Funktion folgen.

Hier ist ein Beispiel von mehreren Konstruktoren zur Initialisierung der Mitglieder der Klasse.

```
//+-----+
//| Klasse um die Namen des Gestalts zu speichern |
//+-----+
class CPerson
{
    string          m_first_name;    // Vorname
    string          m_second_name;   // Familienname
public:
    //-- Ein leerer Default-Konstruktor
    CPerson() {Print(__FUNCTION__);};

    //-- Parametrischer Konstruktor
```

```

        CPerson(string full_name);

    //--- Ein Konstruktor mit der Initialisierungsliste
        CPerson(string surname, string name): m_second_name(surname), m_first_name(name) {}

    void PrintName() {PrintFormat("Name=%s Surname=%s", m_first_name, m_second_name);};
};

//+-----+
//|                                     |
//+-----+
CPerson::CPerson(string full_name)
{
    int pos=StringFind(full_name, " ");
    if(pos>=0)
    {
        m_first_name=StringSubstr(full_name, 0, pos);
        m_second_name=StringSubstr(full_name, pos+1);
    }
};

//+-----+
//| Script program start function      |
//+-----+
void OnStart()
{
    //--- Erhalten eine Fehlermeldung "default constructor is not defined"
    CPerson people[5];
    CPerson Tom="Tom Sawyer";           // Tom Sawyer
    CPerson Huck("Huckleberry", "Finn"); // Huckleberry Finn
    CPerson *Pooh = new CPerson("Winnie", "Pooh"); // Winnie the Pooh
    //--- Zeigen den Wert
    Tom.PrintName();
    Huck.PrintName();
    Pooh.PrintName();

    //--- Das dynamisch erstellten Objekt löschen
    delete Pooh;
};

```

In diesem Fall hat die Klasse CPerson drei Konstruktoren:

1. Expliziter [Default-Konstruktor](#), der Erstellen eines Arrays von Objekten dieser Klasse ermöglicht.
2. Konstruktor mit einem Parameter, der einen vollständigen Namen als Parameter nimmt und ihn auf den Vornamen und Familiennamen nach dem gefundenen Leerzeichen teilt.
3. Konstruktor mit zwei Parametern, der [Initialisierungsliste](#) enthält. Initialisierer - m\_second\_name(surname) und m\_first\_name(name).

Beachten Sie, wie die Initialisierung mit einer Liste Zuweisung ersetzt hat. Einzelne Mitglieder sollten initialisiert werden, wie:

```
Klassenglieder (Liste von Ausdrücken)
```

In der Initialisierungsliste können Mitglieder in beliebiger Reihenfolge gehen, aber alle Mitglieder der

Klasse werden nach der Reihenfolge der Deklaration initialisiert werden. Dies bedeutet, dass in der dritten Konstruktor wird zunächst Mitglied `m_first_name` initialisiert, als er zuerst deklariert war, und nachdem Mitglied `m_second_name` initialisiert wird. Dies sollte in Fällen, in denen die Initialisierung von einigen Klassenmitgliedern von den Werten der anderen Klassenmitglieder hängt, berücksichtigt werden.

Wenn kein Default-Konstruktor in der Basisklasse deklariert ist, und ein oder mehrere Konstruktoren mit Parametern deklariert sind, sollten Sie immer einen der Basisklassenkonstruktoren in der Initialisierungsliste anrufen. Er kommt durch Komma wie übliche Listenmitglieder, und wird zuerst während Initialisierung des Objekts unabhängig von von den Ort in der Initialisierungsliste aufgerufen.

```
//+-----+
//| Basisklasse                                     |
//+-----+
class CFoo
{
    string          m_name;
public:
    //-- Konstruktor mit der Initialisierungsliste
        CFoo(string name) : m_name(name) {Print(m_name);}

};
//+-----+
//| Nachfolger der Klasse CFoo                     |
//+-----+
class CBar : CFoo
{
    CFoo            m_member;          // Klassenglieder ist ein Objekt von Parent
public:
    //-- Default-Konstruktor ruft den Konstruktor von Parent in der Initialisierungsliste
        CBar(): m_member(_Symbol), CFoo("CBAR") {Print(__FUNCTION__);}

};
//+-----+
//| Script program start function                   |
//+-----+
void OnStart()
{
    CBar bar;

;}
```

In diesem Beispiel während der Erstellung des Objekts `bar` wird Default-Konstruktor `CBar()` aufgerufen. In diesem Default-Konstruktor erst wird Konstruktor für den Vorfahr `CFoo`, dann wird Konstruktor für Klassenglied `m_member` aufgerufen.

Destruktor ist eine Sonderfunktion, die automatisch aufgerufen wird, wenn das Klassenobjekt beseitigt wird. Destruktornamen werden so geschrieben, wie der Klassenname mit Tilde (~). Zeilen, dynamische Felder und Objekte, die initialisiert werden müssen, werden auf jeden Fall deinitialisiert unabhängig von der Anwesenheit des Destruktors. Wenn es einen Destruktor gibt, werden diese Handlungen nach Destruktoraufruf durchgeführt werden.

Destruktoren sind immer virtuell, unabhängig davon, ob sie mit dem Schlüsselwort `virtual` erklärt werden oder nicht.



## Methodenbestimmung der Klassen

Funktionen-Methoden der Klasse können sowohl innerhalb, als auch ausserhalb der Klassenerklärung bestimmt werden. Wenn die Methode innerhalb der Klasse bestimmt wird, folgt Ihr Koerper unmittelbar nach der Methodenerklärung.

**Beispiel:**

```
class CTetrisShape
{
protected:
    int          m_type;
    int          m_xpos;
    int          m_ypos;
    int          m_xsize;
    int          m_ysize;
    int          m_prev_turn;
    int          m_turn;
    int          m_right_border;
public:
    CTetrisShape();
    void          SetRightBorder(int border) { m_right_border=border; }
    void          SetYPos(int ypos)         { m_ypos=ypos;           }
    void          SetXPos(int xpos)         { m_xpos=xpos;           }
    int          GetYPos()                   { return(m_ypos);         }
    int          GetXPos()                   { return(m_xpos);         }
    int          GetYSize()                  { return(m_ysize);        }
    int          GetXSize()                  { return(m_xsize);        }
    int          GetType()                   { return(m_type);         }
    void          Left()                     { m_xpos-=SHAPE_SIZE;     }
    void          Right()                    { m_xpos+=SHAPE_SIZE;     }
    void          Rotate()                   { m_prev_turn=m_turn; if(++m_turn>3) m_turn=0; }
    virtual void  Draw()                     { return;                 }
    virtual bool  CheckDown(int& pad_array[]);
    virtual bool  CheckLeft(int& side_row[]);
    virtual bool  CheckRight(int& side_row[]);
};
```

Funktionen mit `SetRightBorder(int border)` nach `Draw()` werden vereinbart und bestimmt innerhalb der Klasse `CTetrisShape`.

Konstruktor `CTetrisShape()` und Methoden `CheckDown(int& pad_array[])`, `CheckLeft(int& side_row[])` und `CheckRight(int& side_row[])` werden nur innerhalb der Klasse erklärt, aber noch nicht bestimmt. Bestimmungen dieser Funktionen muessen weiter nach Kode folgen. Um die Methode ausser der Klasse zu bestimmen, wird [Operation der Kontexterlaubnis](#) verwendet , als Kontext funktioniert der Klassenname.

**Beispiel:**

```

//+-----+
//| Konstruktor der Basisklasse |
//+-----+
void CTetrisShape::CTetrisShape()
{
    m_type=0;
    m_ypos=0;
    m_xpos=0;
    m_xsize=SHAPE_SIZE;
    m_ysize=SHAPE_SIZE;
    m_prev_turn=0;
    m_turn=0;
    m_right_border=0;
}
//+-----+
//| Kontrolle der Moeglichkeit nach unten zu bewegen (für Stab und Kubus) |
//+-----+
bool CTetrisShape::CheckDown(int& pad_array[])
{
    int i,xsize=m_xsize/SHAPE_SIZE;
    for(i=0; i<xsize; i++)
    {
        if(m_ypos+m_ysize>=pad_array[i]) return(false);
    }
    return(true);
}

```

## Zugangsmodifikatoren public, protected und private

Bei der Ausarbeitung der neuen Klasse ist es empfohlen, den Zugang zu Aussengliedern zu begrenzen. Für diese Zwecke werden Schlüsselwoerter **private** oder **protected** verwendet. In diesem Fall wird der Zugang zu diesen versteckten Daten nur aus Funktionen-Methoden derselben Klasse erfolgen. Wenn das Schluesselwort **protected** verwendet wird, kann der Zugang zu versteckten Daten auch aus Methoden der Klassen erfolgen - [Nachfolger](#) dieser Klasse. In gleicher Weise kann der Zugang auch zu Funktionen-Metoden der Klasse begrenzt werden.

Wenn es notwendig ist, den Zugang zu Gliedern und/oder Methoden der Klasse vollkommen zu eroeffnen, wird das Schluesselwort **public** verwendet.

### Beispiel:

```

class CTetrisField
{
private:
    int          m_score;                // Stand
    int          m_ypos;                // laufende Figurlage
    int          m_field[FIELD_HEIGHT][FIELD_WIDTH]; // Dom Matrix
    int          m_rows[FIELD_HEIGHT];  // Numerierung von DOM Reiher
    int          m_last_row;            // Letzte freie Reihe
    CTetrisShape *m_shape;              // Tetrisfigur
    bool         m_bover;               // Spiel beendet
public:
    void         CTetrisField() { m_shape=NULL; m_bover=false; }
}

```

```
void      Init();  
void      Deinit();  
void      Down();  
void      Left();  
void      Right();  
void      Rotate();  
void      Drop();  
private:  
void      NewShape();  
void      CheckAndDeleteRows();  
void      LabelOver();  
};
```

Alle Glieder und Methoden der Klasse, vereinbart nach dem Spezifikator **public:** (und bis folgendem Zugangsspezifikator)), sind zugaenglich bei jeder Referenz zum Objekt dieser Klasse. In diesem Beispiel sind es folgende Glieder: Funktionen CTetrisField(), Init(), Deinit(), Down(), Left(), Right(), Rotate() und Drop().

Alle Glieder der Klasse erklärt nach dem Zugangsspezifikator zu Elementen **private:** (und bis folgendem Zugangsspezifikator) sind nur für die Funktionen-Glieder dieser Klasse zugaenglich. Spezifikatoren des Zugangs zu Elementen beenden immer mit dem Doppelpunkt (:) und können in der Klassenbestimmung mehrmals erscheinen.

Zugang zu Gliedern der Basisklasse kann bei [Vererbung](#) in sekundären Klassen Neubestimmt werden.

Sehen Sie auch

[Objektorientiertes Programmieren](#)

## Objekt des dynamischen Feldes

### Dynamische Felder

Maximum kann ein vierdimensionales Feld zugelassen. Bei der Erklärung des dynamischen Feldes (Feld mit einem nicht angegebenen Wert im ersten Paar der eckigen Klammern) erzeugt der Compiler automatisch eine Variable der oben angegebenen Struktur (Objekt des dynamischen Feldes) und stellt den Code für die richtige Initialisierung bereit.

Dynamische Felder werden automatisch befreit, wenn sie ausserhalb des Blocks gehen, wo sie erklärt werden.

**Beispiel:**

```
double matrix[][10][20]; // dreidimensionales dynamischer Feld
ArrayResize(matrix,5);    // stellen die Größe der ersten Messung
```

### Statische Felder

Bei der expliziten Angabe aller bedeutenden Felddimensionen, verteilt der Compiler im voraus die erforderliche Speichergroesse. Ein solches Feld wird statisch genannt. Dennoch verteilt der Compiler Speicher für Objekt des dynamischen Feldes, das (Objekt) mit dem vorher verteilten Puffer verbunden ist (Speicherteil für Feldaufbewahren).

Erzeugen eines Objekts des dynamischen Feldes ist durch die mögliche Notwendigkeit bedingt, das angegebene statische Feld in eine Funktion als Parameter zu übertragen.

**Beispiele:**

```
double stat_array[5]; // eindimensionales statisches Feld
some_function(stat_array);
...
bool some_function(double& array[])
{
    if(ArrayResize(array,100)<0) return(false);
    ...
    return(true);
};
```

### Felder als Strukturteile

Bei der Erklärung eines statischen Feldes als Glied einer Struktur wird ein Objekt des dynamischen Feldes nicht erzeugt. Das wird für Kompatibilitaet der Datenstrukturen gemacht, die in Windows API verwendet werden.

Aber statische Felder, die als Strukturglieder erklärt werden, können auch in MQL5-Funktionen übertragen werden. In diesem Fall wird bei Parameterübertragung ein zeitliches Objekt des dynamischen Feldes erzeugt werden, das mit dem statischen Feld - Strukturglied verbunden ist.

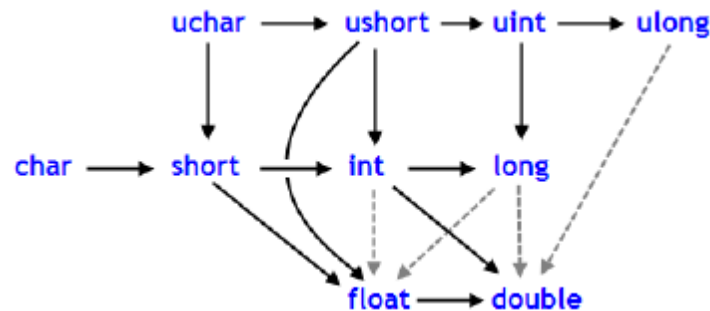
**Sehen Sie auch**

Operationen mit Feldern, Initialisierung der Variablen, Sichtbereich und Lebensdauer der Variablen, Erzeugung und Entfernung der Objekte

## Typenreduzierung

### Reduzierung der Zahlentypen

Oft muss man einen Zahlentyp in den anderen umwandeln. Nicht jeden numerischen Typ kann man in einen anderen umwandeln, zulaessige Reduzierungen in MQL5 werden auf dem Schema gezeigt:



Volllinien mit Richtungszeichen bezeichnen Umwandlungen, die ohne Informationsverlust durchgeführt werden. Anstatt des Typs char kann der Typ [bool](#) auftreten (die beiden nehmen 1 Byte des Speicherplatzes ein), anstatt des Typs int kann der Typ [color](#) verwendet werden (jeder 4 Byte), anstatt des Typs long ist der Typ [datetime](#) zulaessig (jeder 8 Byte). Vier graue Strichlinien auch mit Richtungszeichen bezeichnen Umwandlungen, bei denen der Genauigkeitsverlust entstehen kann. ZB. Zifferzahl in der Ganzzahl 123456789 ([int](#)) ist mehr als es vom Typ [float](#) vertreten werden kann.

```
int n=123456789;
float f=n;      // Inhalt f ist 1.234567892E8
Print("n = ",n,"    f = ",f);
// Ergebnis n= 123456789    f= 123456792.00000
```

Die Zahl, die in Typ float umgewandelt ist, hat dieselbe Ordnung, aber kleinere Genauigkeit. Die den schwarzen Richtungszeichen entgegengesetzte Umwandlungen können mit Informationsverlust durchgeführt werden. Umwandlungen unter char und uchar, short und ushort, int und uint, long und ulong (gemeint werden gegenseitige Umwandlungen), können zum Informationsverlust führen.

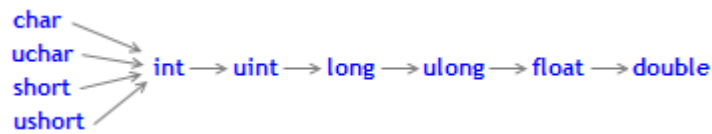
Im Ergebnis der Reduzierung des Fließpunktwertes zum Bruchwert wird der Bruchteil immer weggelassen. Wenn man Fließpunktzahl zur naechstliegenden Ganzzahl runden muss ( in den meisten Faellen ist es nuetzlicher), muss man die Funktion [MathRound\(\)](#) verwenden.

#### Beispiel:

```
//--- Fallbeschleunigung
double g=9.8;
double round_g=(int)g;
double math_round_g=MathRound(g);
Print("round_g = ",round_g);
Print("math_round_g = ",math_round_g);
// Ergebnis:
// round_g = 9
// math_round_g = 10
```

Wenn zwei Werte durch Binaeroperator verbunden werden, wird der Operand des jungeren Typ in den

Operand des älteren Typs umgewandelt laut Priorität zeigt auf dem Schema:



Datentypen char, uchar, short und ushort werden zum Typ int reduziert.

**Beispiele:**

```

char    c1=3;
/---- erstes Beispiel
double d2=c1/2+0.3;
Print("c1/2+0.3 =",d2);
// Ergebnis:   c1/2+0.3 = 1.3

/---- zweites Beispiel
d2=c1/2.0+0.3;
Print("c1/2.0+0.3 =",d2);
// Ergebnis:   c1/2.0+0.3 = 1.8
  
```

Berechnete Ausdruck besteht aus zwei Operationen. Im ersten Beispiel wird die Variable vom ersten Typ char in die zeitweilige Variable des Typs int umgewandelt, denn der zweite Operand in Division, Konstante 2, hat einen älteren Typ int. Im Ergebnis der ganzzahligen Division 3/2 wird der Wert 1, den der Typ int hat.

In zweiter Operation des ersten Beispiels tritt als zweiter Operand Konstante 0.3 auf, die zum Typ double gehört, darum wird das Ergebnis nach der ersten Operation in die zeitweilige Variable des Typs double mit dem Wert 1.0 umgewandelt.

Im zweiten Beispiel wird die Variable vom ersten Typ char in die zeitweilige Variable des Typs double umgewandelt, denn der zweite Operand in Division, Konstante 2.0 gehört zum Typ double; weitere Umwandlungen werden nicht durchgeführt.

## Reduzierung der Zahlentypen

In Ausdrücken der Sprache MQL5 kann explizite und implizite Typenreduzierung verwendet werden. Explizite Typenreduzierung wird auf der Weise geschrieben:

```
var_1 = (Typ)var_2;
```

Als Variable var\_2 kann Ausdruck oder Ergebnis der Funktion sein. Zulaessig ist auch Funktionsaufschreibung der expliziten Typenreduzierung:

```
var_1 = Typ(var_2);
```

Betrachten wir explizite Reduzierung am ersten Beispiel:

```

/---- drittes Beispiel
double d2=(double)c1/2+0.3;
Print("(double)c1/2+0.3 =",d2);
  
```

```
// Ergebnis: (double)c1/2+0.3 = 1.80000000
```

Vor Division wird die Variable `c1` explizit in den Typ `double` umgewandelt. Die derzeit ganzzahlige Konstante 2 wird zum Wert 2.0 des Typs `double` reduziert, denn während der Umwandlungen bekam der erste Operand den Typ `double`. Tatsächlich ist die explizite Typenumwandlung einstellige Operation.

Außerdem kann das Ergebnis bei der Typenreduzierung ausserhalb der zulaessigen Stellenzahl sein. In diesem Fall entsteht Abbruch. ZB.:

```
char c;
uchar u;
c=400;
u=400;
Print("c=",c); // Ergebnis c=-112
Print("u=",u); // Ergebnis u=144
```

Vor den Operationen ( ausser Zuordnungsoperationen) wird die Umwandlung in den Typ durchgeführt, der die groesste Prioritaet hat, und vor den Zuordnungsoperationen - in den Zieltyp.

#### Beispiele:

```
int i=1/2; // keine Typenreduzierung, Ergebnis: 0
Print("i=1/2 ",i);

int k=1/2.0; // Ausdruck wird zum Typ double reduziert,
Print("k=1/2 ",k); // dann zum Zieltyp int reduziert, Ergebnis: 0

double d=1.0/2.0; // keine Typenreduzierung, Ergebnis: 0.5
Print("d=1/2.0; ",d);

double e=1/2.0; // Ausdruck wird zum Typ double reduziert,
Print("e=1/2.0; ",e); // der mit dem Zieltyp zusammenfaellt, Ergebnis: 0.5

double x=1/2; // Ausdruck des Typs int wird zum Zieltyp double reduziert,
Print("x=1/2; ",x); // Ergebnis: 0.0
```

Wenn Sie Typ `long`/`ulong` in `double` konvertieren, kann Genauigkeit verloren gehen: wenn das Ganze ist größer als 9223372036854774784 oder kleiner als -9223372036854774784.

```
void OnStart()
{
    long l_max=LONG_MAX;
    long l_min=LONG_MIN+1;
    //--- Finden wir die maximale ganze Zahl, die Genauigkeit beim Gießen zu double nicht
    while(l_max!=long((double)l_max))
        l_max--;
    //--- Finden wir die kleinste ganze Zahl, die Genauigkeit beim Gießen zu double nicht
    while(l_min!=long((double)l_min))
        l_min++;
    //--- Jetzt zeigen wir das gefundenen Intervall für den ganzen Zahlen
    PrintFormat("Beim Gießen einer ganzen Zahl zu double sollte es "
        "im folgenden Intervall sein [%I64d, %I64d]",l_min,l_max);
}
```



```
//--- Nun wollen wir sehen, was passiert, wenn die Zahl außerhalb dieses Intervalls ist
PrintFormat("l_max+1=%I64d, double(l_max+1)=%.f, ulong(double(l_max+1))=%I64d",
            l_max+1,double(l_max+1),long(double(l_max+1)));
PrintFormat("l_min-1=%I64d, double(l_min-1)=%.f, ulong(double(l_min-1))=%I64d",
            l_min-1,double(l_min-1),long(double(l_min-1)));
//--- erhalten eine solche Schlussfolgerung
// Beim Gießen einer ganzen Zahl zu double sollte es im Intervall [-9223372036854774785; 9223372036854774785] sein
// l_max+1=9223372036854774785, double(l_max+1)=9223372036854774800, ulong(double(l_max+1))=9223372036854774785
// l_min-1=-9223372036854774785, double(l_min-1)=-9223372036854774800, ulong(double(l_min-1))=9223372036854774785
;
```

## Reduzierung für Typ string

Typ string hat die höchste Priorität unter einfachen Typen. Darum wenn einer der Operanden den Typ string hat, wird der andere operand zum Typ string automatisch reduziert. Man muss in Vormerkung behalten, dass es für den Typ string nur einzige zweistellige Operation zulaessig ist. Explizite Reduzierung der Variable des Typs string zum jeden Zahlentyp ist zulaessig.

### Beispiele:

```
string s1=1.0/8;           // Ausdruck wird zum Typ double reduziert,
Print("s1=1.0/8; ",s1);   // dann zum Zieltyp string,
// Ergebnis:"0.12500000"(Zeile mit 10 Zeichen)

string s2=NULL;           // Deinitialisierung der Zeile
Print("s2=NULL; ",s2);    // Ergebnis: Leerzeile
string s3="Ticket N"+12345; // Ausdruck wird zum Typ string reduziert
Print("s=\"Ticket N"+12345,s3);

string str1="true";
string str2="0,255,0";
string str3="2009.06.01";
string str4="1.2345e2";
Print(bool(str1));
Print(color(str2));
Print(datetime(str3));
Print(double(str4));
```

## Typenreduzierung der einfachen Strukturen

Daten des Typs einfache Strukturen können einander zugeordnet werden, wenn alle Glieder beider Strukturen Zahlentypen haben. Dabei müssen in der Zuordnung beide Operanden, links und rechts zum Typ der Strukturen gehören. Es wird nicht Glied-für-Glied Reduzierung durchgeführt, sondern einfache Kopieren. Wenn sich die Strukturgrößen unterscheiden, wird die Bytezahl kopiert, die geringere Größe hat. So wird das Fehlen von Verbindungen (union) in der Sprache MQL5 kompensiert.

### Beispiele:

```

struct str1
{
    double d;
};
//---
struct str2
{
    long l;
};
//---
struct str3
{
    int low_part;
    int high_part;
};
//+-----+
void OnStart()
{
    str1 s1;
    str2 s2;
    str3 s3;
//---
    s1.d=MathArcsin(2.0); // Bekommen eine ungueltige Zahl -1.#IND
    s2=s1;
    printf("1. %f %I64X",s1.d,s2.l);
//---
    s3=s2;
    printf("2. high part of long %.8X low part of long %.8X",
        s3.high_part,s3.low_part);
};

```

Noch ein Beispiel zeigt, wie man eigene Funktion gestalten kann, um aus dem Typ [color](#) Farbdarstellungen in RGB (Red,Green,Blue) zu bekommen. Dafür erzeugen wir zwei Strukturen mit gleicher Größe aber verschiedenem Inhalt. Zur Bequemlichkeit fügen wir in Struktur die Funktion hinzu, die die Farbe in RGB Darstellung in Art der Zeile rückgibt.

```

#property script_show_inputs
input color testColor=clrBlue; // Stellen Sie Farbe für Testen ein
//--- Struktur für Farbdarstellung in RGB
struct RGB
{
    uchar blue; // blaue Farbkomponente
    uchar green; // grüne Farbkomponente
    uchar red; // rote Farbkomponente
    uchar empty; // dieses Byte wird nicht verwendet
    string toString(); // Funktion für Darstellung als Zeile
};
//--- Funktion für Farbausgabe in Form einer Zeile
string RGB::toString(void)

```

```

{
    string out="("+(string)red+": "+(string)green+": "+(string)blue+" )";
    return out;
};
//--- Struktur für Speicherung des abgebildeten Typs color
struct builtColor
{
    color          c;
};
//+-----+
//| Script program start function                               |
//+-----+
void OnStart()
{
    //--- Variable für Speicherung in RGB
    RGB colorRGB;
    //--- Variable für Speicherung des Typs color
    builtColor test;
    test.c=testColor;
    //--- Reduzierung von zwei Strukturen durch Inthaltkopieren
    colorRGB=test;
    Print("color ",test.c,"=",colorRGB.toString());
    //---
};

```

## Reduzierung der Typen von Anzeigern der Basisklassen zu Anzeigern der sekundaeren Klassen

Objekte der [sekundaeren Klasse](#) können auch als Objekte der ihr entsprechenden Basisklasse betrachtet werden. Das führt zu einigen interessanten Folgen. ZB. Gegen die Tatsache, dass Objekte verschiedener Klassen, die von einer Basisklasse entstanden, können sich voneinander unterscheiden, können wir Ihre verbundene Liste erzeugen (List), denn wir betrachten sie als Objekte der Basisklasse. Aber das Gegenteil stimmt nicht: Objekte der Basisklasse sind nicht automatisch Objekte der sekundaeren Klasse.

Man kann explizite Typenreduzierung verwenden, um die Anzeiger der Basisklasse in die [Anzeiger](#) der sekundaeren Klasse umwandeln. Man muss aber ganz sicher sein, dass diese Umwandlung zulaessig ist, sonst entsteht ein kritischer Fehler der Ausführungszeit und das mql5-Programm wird gestoppt.

Sehen Sie auch

[Datentypen](#)

## Typ void und Konstante NULL

Syntaktisch ist der Typ **void** ein Basistyp wie die Typen `char`, `uchar`, `bool`, `short`, `ushort`, `int`, `uint`, `color`, `long`, `ulong`, `datetime`, `float`, `double` und `string`. Dieser Typ wird als Andeutung verwendet, dass die Funktion keinen Wert rückgibt, oder als Funktionsparameter bezeichnet das Fehlen von Parametern.

Vorbestimmte Konstantvariable **NULL** hat den Typ *void*. Sie kann den Variablen jeder anderen Basistypen ohne Umwandlung zugeordnet werden. Zulaessig ist auch der Vergleich von Variablen der Basistypen mit dem Wert **NULL**.

### Beispiel:

```
//--- wenn die Zeile nicht initialisiert wird, ordnen wir ihr unsere vorbestimmte Größe zu  
if(some_string==NULL) some_string="empty";
```

**NULL** kann auch mit Zeigern auf Objekte verglichen werden, erzeugt mit der [Anweisung new](#).

Sehen Sie auch

[Variablen](#), [Funktionen](#)

## Objektanzeiger

In MQL5 ist es möglich, Objekte des komplizierten Typs dynamisch zu erzeugen. Das wird durch [Anweisung new](#) gemacht, die dem Descriptor des erzeugten Objekts zurückgibt. Der Beschreiber kann die Größe 8 Byte sein. Syntaktisch gesehen sind Objekte in MQL5 den Objekten in C++ gleich.

### Beispiele:

```
MyObject* hobject= new MyObject();
```

Wiederholen wir, dass zum Unterschied von C++, ist die Variable *hobject* vom oben angeführten Beispiel **kein Zeiger auf** , sondern Descriptor des Objekts ist.

### Sehen Sie auch

[Variablen](#), [Initialisierung der Variablen](#), [Sichtbereich und Lebensdauer der Variablen](#), [Erzeugung und Entfernung der Objekte](#)

## Referenzen. Modifikator & und Schluesselwort this

### Übertragung der Parameter durch Referenz

In MQL5 können Parameter [einfacher](#) Typen nach dem Wert und durch Referenz übertragen werden, während Parameter [komplizierter](#) Typen werden immer durch Referenz übertragen. Für Andeutung dem Compiler auf die Notwendigkeit, einen Parameter durch Referenz zu übertragen wird vor dem Namen des Parameters das Zeichen Ampersand & gestellt.

Übertragung des Parameters durch Referenz bedeutet die Übertragung der Adresse einer Variable, darum werden alle Veränderungen des übertragenen durch Referenz Parameters werden sofort in der Ausgangsvariable dargestellt. Beim Verwenden von Parameterübertragung durch Referenz kann man gleichzeitige Rückkehr mehrerer Ergebnisse aus der Funktion ermöglichen. Man muss den Modifikator [const](#) verwenden, um der Veränderung des durch Referenz übertragenen Parameters zu.

So wenn der Eingabeparameter der Funktion ein [Feld](#), Objekt einer Struktur oder einer Klasse ist, wird im Header der Funktion nach dem Typ der Variable und vor ihrem Namen das Symbol '&' gestellt.

#### Beispiel

```
class CDemoClass
{
private:
    double          m_array[];

public:
    void            setArray(double &array[]);
};
//+-----+
//|  Feldausfuellung                                     |
//+-----+
void CDemoClass::setArray(double &array[])
{
    if(ArraySize(array)>0)
    {
        ArrayResize(m_array,ArraySize(array));
        ArrayCopy(m_array, array);
    }
}
```

In dem angeführten Beispiel ist die [Klasse](#) CDemoClass erklärt, die ein [privates](#) Glied enthält - Feld m\_array[] des Typs [double](#). Die [Funktion](#) setArray(), in die durch Referenz das Feld array[] übertragen wird, ist erklärt. Wenn Header dieser Funktion ohne Andeutung der Übertragung durch Referenz geschrieben wird, d.h. das Symbol Ampersand wird weggelassen, wird beim Versuch, diesen Code zu compilieren, eine Fehlermeldung erscheinen.

Abgesehen davon, dass das Feld durch Referenz übertragen wird, können wir nicht ein Feld einem anderen Feld zuordnen. Man muss den Inhalt von Feld-Quelle ins Feld-Rezipient elementenweise kopieren. Das Symbol & für Felder und Strukturen bei der Übertragung als Parameter der Funktion ist bei der Funktionsbeschreibung notwendig.

### Schluesselwort this

Variable des Typs Klasse (Objekt) kann sowohl durch Referenz, als auch durch [Anzeiger](#) übertragen

werden. Anzeiger, wie auch Referenz verwendet man, um Zugang zum Objekt zu bekommen. Nach der Erklärung von Objektanzeiger muss man die Anweisung [new](#) für seine Erzeugung und Initialisierung verwenden.

Das reservierte Wort `this` ist für Erhaltung der Referenz des Objektes auf sich selbst bestimmt. Sie ist innerhalb der Klassenmethoden oder Strukturenmethoden zugaenglich. `this` verweist immer auf das Objekt, in dessen Methode es verwendet wird, und der Ausdruck [GetPointer\(this\)](#) gibt Anzeiger des Objektes, dessen Glied die Funktion ist, in der die Funktion `GetPointer()` aufgerufen wurde. In MQL5 können die Funktionen keine Objekte rückgeben, aber sie können Objektanzeiger rückgeben.

D.h. wenn es notwendig ist, dass die Funktion das Objekt rückgibt, koenen wir den Anzeiger dieses Objektes als `GetPointer(this)` rückgeben. Fuegen wir in die Beschreibung der Klasse `CDemoClass` die Funktion `getDemoClass()` hinzu, die Objektanzeiger dieser Klasse rückgibt.

```
class CDemoClass
{
private:
    double          m_array[];

public:
    void            setArray(double &array[]);
    CDemoClass      *getDemoClass();
};
//+-----+
//| Feldausfuellung                                     |
//+-----+
void CDemoClass::setArray(double &array[])
{
    if(ArraySize(array)>0)
    {
        ArrayResize(m_array,ArraySize(array));
        ArrayCopy(m_array,array);
    }
}
//+-----+
//| gibt eigenen Anzeiger zurueck                       |
//+-----+
CDemoClass *CDemoClass::getDemoClass(void)
{
    return(GetPointer(this));
}
```

Strukturen haben keine Anzeiger, man kann die Anweisungen `new` und `delete` zu ihnen nicht anwenden, `GetPointer(this)` kann auch nicht verwendet werden.

Sehen Sie auch

[Objektanzeiger](#), [Erzeugung und Entfernung von Objekten](#), [Sichtbarkeitsbereich und Lebensdauer der Variablen](#)

## Operationen und Ausdrücke

Einigen Symbolen und Symbolfolgen wird ein besonderer Wert gegeben. Das sind sogenannte Operationssymbole, ZB.:

|           |                                    |
|-----------|------------------------------------|
| + - * / % | Symbole arithmetischer Operationen |
| &&        | Symbole logischer Operationen      |
| = += *=   | Symbole der Zuordnungsoperationen  |

Operationssymbole werden in Ausdrücken verwendet und dann sinnvoll, wenn ihnen entsprechende Operanden gegeben sind. Ein besonderer Wert wird auch den Interpunktionszeichen gegeben. Interpunktionszeichen schliessen runde Klammern, Figurklammern, Komma, Doppelpunkt und Semikolon ein.

Operationssymbole, Interpunktionszeichen und Spaces werden verwendet, um Sprachelemente abzugliedern.

In diesem Abschnitt werden folgende Themen betrachtet:

- [Ausdrücke](#)
- [Arithmetische Operationen](#)
- [Zuordnungsoperationen](#)
- [Vergleichsoperationen](#)
- [Logische Operationen](#)
- [Bitorganisierte Operationen](#)
- [Andere Operationen](#)
- [Prioritäten und Operationsordnung](#)



## Ausdrücke

Ausdruck besteht aus einem oder vielen Operanden und Operationssymbolen. Kann in viele Zeile geschrieben werden.

### Beispiele:

```
a++; b = 10;           // mehrere Ausdrücke liegen auf einer Zeile
//--- ein Ausdruck wird in mehrere Zeilen geteilt
x = (y * z) /
    (w + 2) + 127;
```

Ausdruck mit Semikolon beendet (;), ist der Operator.

### Sehen Sie auch

[Prioritäten und Operationsordnung](#)

## Arithmetische Operationen

Zu arithmetischen Operationen gehören additive und Multiplikationsoperationen:

|                                      |                                   |
|--------------------------------------|-----------------------------------|
| Summe von Werten                     | <code>i = j + 2;</code>           |
| Subtrahieren von Werten              | <code>i = j - 3;</code>           |
| Zeichenänderung                      | <code>x = - x;</code>             |
| Multiplikation von Werten            | <code>z = 3 * x;</code>           |
| Quotient einer Division              | <code>i = j / 5;</code>           |
| Rest einer Division                  | <code>minutes = time % 60;</code> |
| Addieren von 1 zum Variablenwert     | <code>i++;</code>                 |
| Addieren von 1 zum Variablenwert     | <code>++i;</code>                 |
| Subtrahieren von 1 vom Variablenwert | <code>k--;</code>                 |
| Subtrahieren von 1 vom Variablenwert | <code>--k;</code>                 |

Inkrement- und Dekrementoperationen werden nur Variablen gegenüber verwendet, nicht Konstanten gegenüber. Präfixinkrement (`++i`) und Präfixdekrement (`--k`) werden variable gegenüber unmittelbar vor der Verwendung dieser Variable im Ausdruck.

Postfixinkrement (`i++`) und Postfixdekrement (`k--`) werden der Variable gegenüber verwendet sofort nach der Verwendung dieser Variable im Ausdruck.

### Wichtiger Hinweis

```
int i=5;
int k = i++ + ++i;
```

Rechnerische Probleme können auftreten, während Sie den obigen Ausdruck von einer Programmierumgebung zu einem anderen (wie zB von Borland C++ in MQL5). Im Allgemeinen hängt die Reihenfolge der Auswertung auf der Compilerimplementierung. In der Praxis gibt es zwei Möglichkeiten, um Post-Dekrement (Post-Inkrement) zu implementieren:

1. Die Post-Dekrement (Post-Inkrement) wird der Variablen nach der Berechnung der ganze Ausdruck angewendet;
2. Die Post-Dekrement (Post-Inkrement) wird der Variablen sofort bei der Operation angewendet.

In MQL5 derzeit ist die erste Methode zur Berechnung der Post-Dekrement (Post-Inkrement) implementiert. Aber selbst mit diesem Wissen ist es besser mit der Nutzung dieser Subtilität nicht zu experimentieren.

### Beispiele:

```
int a=3;
a++;           // richtiger Ausdruck
int b=(a++)*3; // falscher Ausdruck
```

### Sehen Sie auch

[Prioritäten und Operationsordnung](#)

## Zuordnungsoperationen

Der Ausdruckswert, dessen Teil eine Zuordnungsoperation ist, der Wert des linken Operanden nach der Zuordnung:

|                                      |                     |
|--------------------------------------|---------------------|
| Zuordnung der Größe x der Variable y | <code>y = x;</code> |
|--------------------------------------|---------------------|

Folgende Operationen verbinden arithmetische oder bitorganisierte Operationen mit der Zuordnungsoperation:

|                                                                                |                             |
|--------------------------------------------------------------------------------|-----------------------------|
| Anstieg der Variable y um x                                                    | <code>y += x;</code>        |
| Erniedrigung der Variable y um x                                               | <code>y -= x;</code>        |
| Multiplikation der Variable y mit x                                            | <code>y *= x;</code>        |
| Division der Variable y durch x                                                | <code>y /= x;</code>        |
| Rest der Division der Variable y durch x                                       | <code>y %= x;</code>        |
| Verschiebung der Binaerdarstellung y rechts um x Bit                           | <code>y &gt;&gt;= x;</code> |
| Verschiebung der Binaerdarstellung y links um x Bit                            | <code>y &lt;&lt;= x;</code> |
| Bitorganisierte Operation UND der Binaerdarstellungen y und x                  | <code>y &amp;= x;</code>    |
| Bitorganisierte Operation ODER der Binaerdarstellungen y und x                 | <code>y  = x;</code>        |
| Bitorganisierte Operation ausschliessendes ODER<br>Binaerdarstellungen y und x | <code>y ^= x;</code>        |

Bitorganisierte Operationen werden nur mit Ganzzahlen durchgeführt. Bei der Operationsdurchführung logische Verschiebung der Darstellung y rechts/links um x Bit werden 5 Binaerstellen mit niedrigem Stellenwert des Wertes x verwendet, höherwertige Stellen werden weggelassen, d.h. die Verschiebung wird um 0-31 Bit durchgeführt.

Bei der Operationsdurchführung %= (Wert y des Moduls x) Ergebniszeichen faellt mit dem Zeichen des Dividenden zusammen.

Im Ausdruck kann der Zuordnungsoperator mehrmals auftreten. In diesem Fall wird die Ausdrucksbearbeitung von rechts nach links durchgeführt:

|                     |
|---------------------|
| <code>y=x=3;</code> |
|---------------------|

Zuerst wird der Variable x der Wert 3 zugeordnet, dann wird der Variable y der Wert von Variable x zugeordnet, d.h. auch 3.

Sehen Sie auch

[Prioritaeten und Operationsordnung](#)

## Vergleichsoperationen

Logischer Wert FALSCH wird von Null-Wert dargestellt, und der Wert RICHTIG wird von jedem Nichtnull-Wert dargestellt.

Der Wert von Ausdrücken, die Vergleichsoperationen oder [logische Operationen](#) enthalten, sind FALSCH(0) oder RICHTIG(1).

|                                            |                         |
|--------------------------------------------|-------------------------|
| Richtig, wenn a gleich b                   | <code>a == b;</code>    |
| Richtig, wenn a ungleich b                 | <code>a != b;</code>    |
| Richtig, wenn a kleiner als b              | <code>a &lt; b;</code>  |
| Richtig, wenn a groesser als b             | <code>a &gt; b;</code>  |
| Richtig, wenn a kleiner als oder b         | <code>a &lt;= b;</code> |
| Richtig, wenn a groesser als oder gleich b | <code>a &gt;= b;</code> |

Man kann nicht zwei [Realzahlen](#) vergleichen. In meisten Faellen können zwei annäherlich gleiche Zahlen nicht gleich sein wegen des Unterschiedes im 15. Zeichen nach dem Komma. Für korrekten Vergleich der zwei Realzahlen muss man normalisierte Differenz dieser Zahlen mit dem Nullwert vergleichen.

**Beispiel:**

```
bool CompareDoubles(double number1, double number2)
{
    if(NormalizeDouble(number1-number2, 8) == 0) return(true);
    else return(false);
};}

void OnStart()
{
    double first=0.3;
    double second=3.0;
    double third=second-2.7;
    if(first!=third)
    {
        if(CompareDoubles(first, third))
            printf("%.16f  %.16f  dennoch sind sie gleich ", first, third);
    }
};}

// Ergebnis: 0.3000000000000000  0.2999999999999998  dennoch gleich
```

Sehen Sie auch

[Prioritaeten und Operationsordnung](#)

## Logische Operationen

### Logische Negation NICHT(!)

Operand der logischen Negation NICHT (!) muss den arithmetischen Typ haben. Das Ergebnis ist RICHTIG(1), wenn der Wert des Operanden FALSCH(0) ist, und FALSCH(0), wenn der Operand nicht gleich FALSCH(0) ist).

```
if(!a) Print("nicht 'a'");
```

### Logische Operation ODER (||)

Logische Operation ODER (||) der Werte x und y. Der Ausdruckswert ist RICHTIG(1), wenn der Wert von x oder y gueltig (nicht Null) ist. Im Gegenfall - FALSCH (0).

```
if(x<0 || x>=maxBars) Print("out of range");
```

### Logische Operation UND (&&)

Logische Operation UND (&&) der Werte x und y. Der Ausdruckswert ist RICHTIG (1), wenn die Werte x und y gueltig sind (nicht Null). Im Gegenfall - FALSCH(0). Logische Ausdrücke werden vollkommen berechnet, d.h. nicht mit dem Schema "Kurzberechnung".

```
if(p!=x && p>y) Print("TRUE");
```

### Kurze Einschätzung der logischen Operationen

Logischen Ausdrücken gegenüber wird das Schema der sogenannten "kurzen Einschätzung" angewendet ,d.h. Berechnung des Ausdrucks wird dann beendet, wenn man das Ergebnis des Ausdrucks genau einschätzen kann.

```
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//--- das erste Beispiel der kurzen Einschätzung
if(func_false() && func_true())
{
Print("Operation &&: Diese Meldung sehen Sie nie");
};
else
{
Print("Operation &&: Das Ergebnis des ersten Ausdrucks false, darum wurde der zw
};
//--- das zweite Beispiel der kurzen Einschätzung
if(!func_false() || !func_true())
{
Print("Operation ||: Das Ergebnis des ersten Ausdrucks true, darum wurde der zwe
```

```
    ;}
    else
    {
        Print("Operation ||: Diese Meldung sehen Sie nie");
    }
    ;}
; }

//+-----+
//| Funktion gibt immer false zurück |
//+-----+
bool func_false()
{
    Print("Funktion func_false()");
    return(false);
; }

//+-----+
//| Funktion gibt immer true zurück |
//+-----+
bool func_true()
{
    Print("Funktion func_true()");
    return(true);
; }
```

Sehen Sie auch

[Prioritaeten und Operationsordnung](#)

## Bitorganisierte Operationen

### 1-Komplement

1-Komplement der Variable. Ausdruckswert hat 1 in allen Stellen, wo die Variable 0 hat und 0 in allen Stellen, wo die Variable 1 hat.

```
b = ~n;
```

**Beispiel:**

```
char a='a',b;  
b=~a;  
Print("a = ",a, " b = ",b);  
// Ergebnis wird derartig sein:  
// a = 97 b = -98
```

### Verschiebung nach rechts

Binaerdarstellung x wird nach rechts um y Stellen verschoben. Wenn der verschobene Wert den zeichenlosen Typ hat, wird die logische Verschiebung nach rechts durchgeführt, d.h. die sich links befreiende Stellen werden mit Nullen ausgefüllt werden.

Wenn aber der verschobene Ausdruck einen Zeichentyp hat, wird arithmetische Verschiebung nach rechts durchgeführt, d.h. die sich links befreiende Stellen werden von Zeichenbits ausgefüllt ( wenn die Zahl positiv ist, wird der Wert von Zeichenwert 0 sein, wenn die Zahl negativ ist, wird der Wert von Zeichenbit 1 sein).

```
x = x >> y;
```

**Beispiel:**

```
char a='a',b='b';  
Print("Before: a = ",a, " b = ",b);  
//--- verschieben wir nach rechts  
b=a >> 1;  
Print("After: a = ",a, " b = ",b);  
// Ergebnis wird derartig sein:  
// Before: a = 97 b= 98  
// After: a = 97 b= 48
```

### Verschiebung nach links

Binaerdarstellung x wird nach links um y Stellen verschoben; sich befreiende Stellen werden von Nullen ausgefüllt.

```
x = x << y;
```

**Beispiel:**

```
char a='a',b='b';
```

```
Print("Before:  a = ",a, "  b = ",b);
//--- verschieben wir nach links
b=a << 1;
Print("After:   a = ",a, "  b = ",b);
// Ergebnis wird derartig sein:
// Before:  a = 97   b = 98
// After:   a = 97   b = -62
```

Es ist nicht empfehlenswert, um mehrere oder gleiche Anzahl der Bits zu verschieben, als die Laenge der verschobten Variable, denn das Ergebnis der Operation ist unbestimmt.

## Bitorganisierte Operation UND

Bitorganisierte Operation UND der Binaerdarstellungen x und y. Der Ausdruckswert hat 1 (RICHTIG) in allen Stellen, in denen x und y keine 0 haben; und 0 (FALSCH) - in allen anderen Stellen.

```
b = ((x & y) != 0);
```

Beispiel:

```
char a='a',b='b';
//--- Operation UND
char c=a&b;
Print("a = ",a,"  b = ",b);
Print("a & b = ",c);
// Ergebnis wird derartig sein:
// a = 97   b = 98
// a & b = 96
```

## Bitorganisierte Operation ODER

Bitorganisierte Operation ODER der Binaerdarstellungen x und y. Ausdruckswert hat 1 in allen Stellen ,wo x oder y keine 0 hat, und 0 - in allen anderen Stellen.

```
b = x | y;
```

Beispiel:

```
char a='a',b='b';
//--- Operation ODER
char c=a|b;
Print("a = " ,a,"  b = ",b);
Print("a | b = ",c);
// Ergebnis wird derartig sein:
// a = 97   b = 98
// a | b = 99
```

## Bitorganisierte Operation ausschliessendes ODER



Bitorganisierte Operation ausschliessendes ODER (eXclusive OR) der Binaerdarstellungen x und y. Der Ausdruckswert hat 1 in den Stellen, in denen x und y verschiedene Binaerwerte haben, und 0 - in allen anderen Stellen.

```
b = x ^ y;
```

**Beispiel:**

```
char a='a', b='b';  
//--- Operation ausschliessendes ODER  
char c=a^b;  
Print("a = ",a," b = ",b);  
Print("a ^ b = ",c);  
// Ergebnis wird derartig sein:  
// a = 97   b = 98  
// a ^ b = 99
```

Bitorganisierte Operationen werden nur mit [Ganzzahlen](#) durchgeführt.

**Sehen Sie auch**

[Prioritaeten und Operationsordnung](#)

## Andere Operationen

### Indizieren ( [ ] )

Beim Zugriff zum i-Feldelement ist der Ausdruckswert der Wert von Variable mit Laufnummer 1.

**Beispiel:**

```
array[i] = 3; // dem i-Element des Feldes array den Wert 3 zuordnen.
```

Feldindex kann nur Ganzzahl sein. Zugelassen werden nicht mehr als vierdimensionale Felder. Indizieren jeder Messung wird von 0 bis der Messungsgroesse -1 durchgeführt. Beim eindimensionalen Feld aus 50 Elementen wird der Zugriff zum ersten Element als array [0] aussehen, zum letzten Element - array[49].

Beim Zugriff ausserhalb des Feldes wird das Subsystem einen kritischen Fehler generieren und Durchführung des Programms wird unterbrochen.

### Funktionsaufruf mit Argumenten x1, x2,..., xn

Jedes Argument kann Konstante, Variable oder Ausdruck des entsprechenden Typs darstellen. Übergabeargumente werden durch Kommas getrennt und muessen sich innerhalb runder Klammern befinden, linke Klammer muss nach den Namen der Aufrufunktion folgen.

Ausdruckswert ist der Wert, der durch die Funktion zurückgegeben wird. Wenn der Typ des Rückgabewertes einer Funktion void ist, kann der Aufruf dieser Funktion nicht rechts in der Zuordnungsoperation gestellt werden. Seien Sie bitte sicherm, dass Durchführungsordnung der Ausdrücke x1,..., xn garantiert wird .

**Beispiel:**

```
int length=1000000;
string a="a",b="b",c;
//---
int start=GetTickCount(),stop;
long i;
for(i=0;i<length;i++)
{
    c=a+b;
};
stop=GetTickCount();
Print("time for 'c = a + b' = ",(stop-start)," milliseconds, i = ",i);
```

### Operation Komma ( , )

Die durch Kommas getrennten Ausdrücke werden von links nach rechts berechnet. Alle Nebeneffekte des linken Ausdrucks können vor der Berechnung des rechten Ausdrucks erscheinen. Typ und Ergebniswert faellt mit dem Typ und Ergebniswert des rechten Ausdrucks zusammen. Als Beispiel kann

die Liste der übertragenen Parameter gesehen werden (s. oben).

**Beispiel:**

```
for(i=0,j=99; i<100; i++,j--) Print(array[i][j]);
```

## Operation Punkt ( . )

für direkten Zugriff zu öffentlichen Gliedern der Strukturen und Klassen wird die Operation Punkt verwendet. Syntax

```
Name der Variable des Typs Strukturen .Name des Gliedes
```

**Beispiel:**

```
struct SessionTime
{
    string sessionName;
    int    startHour;
    int    startMinutes;
    int    endHour;
    int    endMinutes;
} st;
st.sessionName="Asian";
st.startHour=0;
st.startMinutes=0;
st.endHour=9;
st.endMinutes=0;
```

## Operation mit Kontexterlaubnis ( :: )

Jede Funktion im mql5-Programm hat ihr eigenes Durchführungskontext. ZB. die Systemfunktion [Print\(\)](#) wird im Globalkontext durchgeführt. [Importfunktionen](#) werden im Kontext des entsprechenden Imports aufgerufen. Funktionen-Methoden der [Klassen](#) haben Kontext der entsprechenden Klasse. Syntax der Operation der Kontextfreigabe:

```
[Kontextname]::Funktionsname(Parameter)
```

Wenn der Kontextname fehlt, ist es explizites Zeichen vom Verwenden des globalen Kontextes. Wenn die Operation der Kontextfreigabe fehlt, wird die Funktion im nächsten Kontext gesucht. Wenn die Funktion im lokalen Kontext fehlt, wird die Suche im globalen Kontext durchgeführt.

Die Operation der Kontextfreigabe wird auch für Definieren der Funktion-Glieder einer Klasse verwendet.

```
Typ Klassenname::Funktionsname(Parameterbeschreibung)
{
    // Funktionskörper
};
```

Verwendung von mehreren Funktionen mit dem gleichen Namen von verschiedenen Ausführungskontexte in einem Programm kann Mehrdeutigkeit führen. Die Prioritätsreihenfolge der Funktionsaufrufe ohne ausdrückliche Spezifikation von Umfang ist wie folgt:

1. Klassenmethoden. Wenn es keine Funktion mit dem angegebenen Namen in der Klasse gibt, gehen Sie zum nächsten Level.
2. MQL5-Funktionen. Wenn die Sprache nicht eine solche Funktion hat, gehen Sie zum nächsten Level.
3. Benutzerdefinierte globale Funktionen. Wenn keine Funktion mit solchem Namen gefunden ist, gehen Sie zum nächsten Level.
4. Importierte Funktionen. Wenn keine Funktion mit dem angegebenen Namen gefunden wird, gibt der Compiler einen Fehler zurück.

Um die Mehrdeutigkeit der Funktionsaufrufe zu vermeiden, geben Sie immer den Funktionsumfang mit Hilfe der Kontexterlaubnisoperationen.

#### Bespiel:

```
#property script_show_inputs
#import "kernel32.dll"
    int GetLastError(void);
#import

class CCheckContext
{
    int          m_id;
public:
    CCheckContext() { m_id=1234; }

protected:
    int          GetLastError() { return(m_id); }
};

class CCheckContext2 : public CCheckContext
{
    int          m_id2;
public:
    CCheckContext2() { m_id2=5678; }

    void          Print();
protected:
    int          GetLastError() { return(m_id2); }
};

void CCheckContext2::Print()
{
    ::Print("Terminal GetLastError", ::GetLastError());
    ::Print("kernel32 GetLastError", kernel32::GetLastError());
    ::Print("parent GetLastError", CCheckContext::GetLastError());
    ::Print("our GetLastError", GetLastError());
}

//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    //---
    CCheckContext2 test;
    test.Print();
}

//+-----+
```

## Operation der Größenentnahme des Datentyps oder der Größe des Objektes jedes Datentyps ( sizeof )

Mittels der Operation `sizeof` kann die Speichergroesse bestimmt werden, die dem Identifikatoren oder dem Typ entspricht. Operation `sizeof` hat folgendes Format:

**Beispiel:**

```
sizeof (Ausdruck)
```

Als Ausdruck kann jeder Identifikator oder Typenname in Klammern verwendet werden. Merken wir an, dass der Typenname `void` nicht verwendet werden kann, und Identifikator kann nicht zum Bitfeld gehören oder Funktionsname sein.

Wenn als Ausdruck der Name des statistischen Feldes angegeben wird ( d.h. die erste Dimensionierung vorgegeben wird), wird als Ergebnis die Größe des ganzen Feldes sein (d.h. Produkt der Multiplikation der Elementenzahl mit Typenlaenge). wenn als Ausdruck der Name des dynamischen Feldes angegeben wird (erste Dimensionalisierung nicht vorgegeben), wird als Ergebnis die Größe des Objektes des dynamischen Feldes.

Wenn `sizeof` zum Strukturnamen oder Klassennamen gegenüber verwendet wird oder dem Identifikatoren, der den Typ des Struktur oder Klasse hat, wird als Ergebnis die tatsaechliche Größe der Struktur oder Klasse sein.

**Beispiel:**

```
struct myStruct
{
    char    h;
    int     b;
    double  f;
} str;

Print("sizeof(str) = ", sizeof(str));
Print("sizeof(myStruct) = ", sizeof(myStruct));
```

Größenberechnung wird in der Etappe der Compilierung durchgeführt.

**Sehen Sie auch**

[Prioritaeten und Operationsordnung](#)

## Prioritaeten und Operationsordnung

für jede Gruppe der Operationen ist die Prioritaet in der Tabelle gleich. Je hoehe Prioritaet von Gruppe der Operationen ist, desto hoeher sich die Gruppe in der Tabelle befindet. Reihenfolge der Befehle bestimmt Gruppieren von Operationen und Operanden.

**Achtung:** Prioritaet der Operationsdurchfuehrung in Sprache MQL5 entspricht der Prioritaet in Sprache C++, und unterscheidet sich von der Prioritaet vorgegeben in MQL4.

| Operation                                  | Beschreibung                                                                                                                                                                                                         | Reihenfolge der Befehle |
|--------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------|
| ()<br>[]<br>.                              | Funktionsaufruf<br>Hervorheben des<br>Feldelementes<br>Hervorheben des<br>Strukturelementes                                                                                                                          | Von links nach rechts   |
| !<br>~<br>-<br>++<br>--<br>(Typ)<br>sizeof | Logische Verneinung<br>Bitorganisierte Verneinung<br>(complement)<br>Zeichenänderung<br>Erhoehung um eins<br>(increment)<br>Erniedrigung um eins<br>(decrement)<br>Typumwandlung<br>Bestimmung der Größe in<br>Bytes | von rechts nach links   |
| *<br>/<br>%                                | Multiplikation<br>Division<br>Division modulo                                                                                                                                                                        | von links nach rechts   |
| +<br>-                                     | Addition<br>Subtrahieren                                                                                                                                                                                             | von links nach rechts   |
| <<<br>>>                                   | Verschiebung nach links<br>Verschiebung nach rechts                                                                                                                                                                  | von links nach rechts   |
| <<br><=<br>><br>>=                         | kleiner als<br>kleiner als oder gleich<br>groesser als<br>groesser als oder gleich                                                                                                                                   | von links nach rechts   |
| ==<br>!=                                   | Gleich<br>Ungleich                                                                                                                                                                                                   | von links nach rechts   |
| &                                          | Bitorganisierte Operation UND                                                                                                                                                                                        | von links nach rechts   |
| ^                                          | Bitorganisierte Operation<br>ausschliessendes ODER<br>(eXclude OR)                                                                                                                                                   | von links nach rechts   |
|                                            | Bitorganisierte operation<br>ODER                                                                                                                                                                                    | von links nach rechts   |

|                                                                  |                                                                                                                                                                                                                                                                                                                                                                                       |                       |
|------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|
| &&                                                               | Logische operation UND                                                                                                                                                                                                                                                                                                                                                                | von links nach rechts |
|                                                                  | Logische Operation ODER                                                                                                                                                                                                                                                                                                                                                               | von links nach rechts |
| ?:                                                               | Bedingungsoperation                                                                                                                                                                                                                                                                                                                                                                   | Von rechts nach links |
| =<br>*=<br>/=<br>%=<br>+=<br>-=<br><=<=<br>>>=<br>&=<br>^=<br> = | Zuordnung<br>Multiplikation mit Zuordnung<br>Division mit Zuordnung<br>Division modulo<br>Addition mit Zuordnung mit<br>Zuordnung<br>Subtrahieren mit Zuordnung<br>Verschiebung nach links mit<br>Zuordnung<br>Verschiebung nach rechts mit<br>Zuordnung<br>Bitorganisiertes UND mit<br>Zuordnung<br>Ausgeschlossenes ODER mit<br>Zuordnung<br>Bitorganisiertes ODER mit<br>Zuordnung | Von rechts nach links |
| ,                                                                | Komma                                                                                                                                                                                                                                                                                                                                                                                 | Von links nach rechts |

für die Veränderung der Reihenfolge werden runde Klammern verwendet, die die hoechste Prioritaet haben.

## Operatoren

Operatoren der Sprache beschreiben einige algorithmische Handlungen, die man für die Aufgabenloesung durchführen muss. Programmkörper ist die Folge solches Operators. Die aufeinander folgenden Operatoren werden durch Semikolon getrennt.

| Operator                              | Beschreibung                                                                                                                                                                                       |
|---------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <u>Zusammengesetzter Operator</u> { } | eine oder mehrere Operatoren des Typs in geschweiften Klammern { }                                                                                                                                 |
| <u>Operator-Ausdruck</u> ( ; )        | Jeder Ausdruck der mit Semikolon beendet ( ; )                                                                                                                                                     |
| Operator <u>return</u>                | Unterbricht die laufende Funktion und gibt die Steuerung dem Aufrufprogramm zurück.                                                                                                                |
| Konditionaloperator <u>if-else</u>    | Verwendet bei der Notwendigkeit, die Wahl zu treffen                                                                                                                                               |
| Wenn-Anweisung <u>?:</u>              | Einfacherer Analog der Anweisung if-else                                                                                                                                                           |
| Auswahanweisung <u>switch</u>         | Gibt die Steuerung der Anweisung, die dem Ausdruckswert entspricht                                                                                                                                 |
| Zyklusweisung <u>while</u>            | Anweisung wird durchgeführt, bis der Ausdruck falsch ist. Ausdruckspruefung wird vor jeder Iteraion durchgeführt.                                                                                  |
| Zyklusweisung <u>for</u>              | Anweisung wird durchgeführt, bis der Ausdruck falsch ist. Ausdruckspruefung wird vor jeder Iteraion durchgeführt                                                                                   |
| Zyklusweisung <u>do-while</u>         | Anweisung wird durchgeführt, bis der Ausdruck falsch ist. Pruefung der Schlussbedingung wird am Ende, nach jedem Zyklusgang durchgeführt. Zykluskoerper wird immer wenigstens einmal durchgeführt. |
| Anweisung <u>break</u>                | Unterbricht die Durchführung der naechsten eingebetteten Aussenanweisung switch, while, do-while oder for                                                                                          |
| Anweisung <u>continue</u>             | Gibt die Steuerung am Ende der naechsten Aussenzyklusweisung while, do-while oder for                                                                                                              |
| Anweisung <u>new</u>                  | Erzeugt das Objekt entsprechender Größe und gibt Descriptor des erzeugten Objektes zurück.                                                                                                         |
| Anweisung <u>delete</u>               | Entfernt das von Anweisung new geschaffenes Objekt                                                                                                                                                 |

Anweisung kann eine oder mehrere Zeilen besetzen. Zwei oder mehrere Anweisungen können sich auf einer Zeile befinden. Anweisungen, Die die Reihenfolge der Befehle regeln (if, if-else, switch, while und for) können in einander eingebettet sein.

**Beispiel:**



```
if(Month() == 12)
    if(Day() == 31) Print("Happy New Year!");
```

**Sehen Sie auch**

[Initialisierung der Variablen](#), [Sichtbereich und Lebensdauer der Variablen](#), [Erzeugung und Entfernung der Objekte](#)

## Zusammengesetzter Operator

Zusammengesetzter Operator (Block) besteht aus einem oder mehreren Operatoren jedes Typs in geschweiften Klammern { }. Nach der rechten geschweiften Klammer muss kein Semikolon stehen (;).

Beispiel:

```
if(x==0)
{
    Print("invalid position x=",x);
    return;
}
```

Sehen Sie auch

[Initialisierung der Variablen](#), [Sichtbereich und Lebensdauer der Variablen](#), [Erzeugung und Entfernung der Objekte](#)

## Operator-Ausdruck

Jeder Ausdruck, der mit Semikolon (;) beendet, ist ein Operator. Weiter folgen Beispiele von Operatoren-Ausdrücke.

### Operator der Zuordnung:

Identifikator = Ausdruck;

```
x=3;  
y=x=3;  
bool equal=(x==y);
```

Im Ausdruck kann Operator der Zuordnung mehrmals auftreten. In diesem Fall wird die Ausdrucksbearbeitung von rechts nach links durchgeführt.

### Operator des Funktionsaufrufs:

Funktionsname (Argument1,..., ArgumentN);

```
FileClose(file);
```

### Leerer Operator

Besteht ausschliesslich aus Semikolon (;) und wird verwendet für die Bezeichnung von Leerkooper der Steueranweisung.

Sehen Sie auch

[Initialisierung der Variablen](#), [Sichtbereich und Lebensdauer der Variablen](#), [Erzeugung und Entfernung der Objekte](#)

## Rücksetzoperator return

Operator return stoppt die laufende [Funktion](#) und gibt Steuerung dem Aufrufprogramm zurück. Ergebnis der Berechnung wird der Aufruffunktion zurückgegeben. Ausdruck kann einen Operator der Zuordnung haben.

**Beispiel:**

```
int CalcSum(int x, int y)
{
    return(x+y);
};
```

In Funktionen mit dem Typ des Rücksatzwertes [void](#) muss man den Operator [return](#) ohne Ausdruck verwenden:

```
void SomeFunction()
{
    Print("Hello!");
    return;    // dieser Operator kann entfernt werden.
};
```

Rechte geschweifte Klammer setzt explizite Durchführung des Operators [return](#) ohne Ausdruck voraus.

Man kann [einfache Typen](#), [einfache Strukturen](#), [Objektanzeiger](#) rückgeben. Durch den Operator *return* können keine Felder, Klassenobjekte, Variablen der komplizierten Strukturen zurückgegeben werden.

**Sehen Sie auch**

[Initialisierung der Variablen](#), [Sichtbereich und Lebensdauer der Variablen](#), [Erzeugung und Entfernung der Objekte](#)

## Konditionaloperator if-else

Operator IF - ELSE wird bei der Notwendigkeit, eine Wahl zu treffen, verwendet. Vorschriftmaessig sieht die Syntax so aus:

```
if (Ausdruck)
    Operator1
else
    Operator2
```

Wenn der Ausdruck richtig ist, wird der Operator1 durchgeführt und die Steuerung wird an den Operator gegeben, die dem Operator2 folgt (d.h. die Operator2 wird nicht durchgeführt). Wenn der Ausdruck falsch ist, wird die Operator2 durchgeführt.

Teil **else** des Operators **if** kann weggelassenwerden. Darum kann in eingebetteten Operatoren **if** mit weggelassenem Teil **else** Mehrdeutigkeit entstehen. In diesem Fall verbindet sich **else** mit der naechsten vorangehenden Operator **if** in demselben Block, die keinen Teil **else** hat.

### Beispiele:

```
//--- Teil else gehört zum zweiten Operator if:
if(x>1)
    if(y==2) z=5;
else      z=6;
//--- Teil else gehört zum ersten Operator if
if(x>1)
{
    if(y==2) z=5;
; }
else      z=6;
//--- Eingebettete Operatoren
if(x=='a')
{
    y=1;
; }
else if(x=='b')
{
    y=2;
    z=3;
; }
else if(x=='c')
{
    y=4;
; }
else Print("ERROR");
```

### Sehen Sie auch

[Initialisierung der Variablen](#), [Sichtbereich und Lebensdauer der Variablen](#), [Erzeugung und Entfernung der Objekte](#)

## Ternärer Operator ?:

Gesamtform des ternären Operators sieht so aus:

```
Ausdruck1? Ausdruck2:Ausdruck3
```

Als erster Operand - "Ausdruck1" - kann jeder Ausdruck verwendet werden, dessen Ergebnis der Wert [bool](#) ist. Wenn das Ergebnis [true](#) ist, wird die Anweisung durchgeführt, die vom zweiten Operator vorgegeben wird, d.h. "der Ausdruck2".

Wenn aber der erste Operand [false](#) wird der dritte Operand - "Ausdruck3" durchgeführt. Der zweite und der dritte Operand müssen Werte eines Typs zurückgeben und den Typ [void](#) nicht haben. Ergebnis der Durchführung des Wenn-Operators ist das Ergebnis des Ausdrucks2 oder Ergebnis des Ausdrucks3, abhängig vom Ergebnis des Ausdrucks1.

```
//--- normieren wir Differenz unter Eröffnungspreisen und Schlusspreisen im Tagsausmass
double true_range = (High==Low)?0:(Close-Open)/(High-Low);
```

Diese Aufzeichnung ist der nächsten äquivalent

```
double true_range;
if(High==Low)true_range=0; // wenn High und Low gleich sind
else true_range=(Close-Open)/(High-Low); // wenn Ausmass nicht Null ist
```

## Einschränkungen bezüglich der Verwendung des Operators

Basierend auf dem Wert von "Ausdruck1", muss der Operator einen der zwei Werte zurückgeben - entweder "Ausdruck2" oder "Ausdruck3". Es gibt einige Einschränkungen auf diese Ausdrücke:

1. Verwechseln Sie nicht den [benutzerdefinierten Typ](#) mit dem [einfachen Typ](#) oder [Aufzählung](#). [NULL](#) kann für den [Zeiger](#) verwendet werden.
2. Wenn Typen von Werten einfach sind, wird der Operator des maximalen Typs sein (siehe [Typ-Umwandlung](#)).
3. Wenn einer der Werte eine Aufzählung ist und die zweite von einem numerischen Typ ist, wird die Aufzählung durch int ersetzt und die zweite Regel angewendet wird.
4. Wenn beide Werte Aufzählungen sind, müssen ihre Typen identisch sein, und der Operator wird vom Typ Aufzählung sein.

Einschränkungen für die benutzerdefinierte Typen (Klassen oder Strukturen):

- a) Typen müssen identisch sein oder sollte eine von dem anderen [abgeleitet](#) werden.
- b) Wenn Typen nicht identisch sind (Vererbung), dann wird das Kind implizit zum übergeordneten Typ umgewandelt, dh der Operator wird von der übergeordneten Typ sein.
- c) Verwechseln Sie nicht das Objekt und den Zeiger - beide Ausdrücke sind entweder Objekte oder [Zeiger](#). [NULL](#) kann für den Zeiger verwendet werden..

### Bemerkung

Seien Sie vorsichtig bei der Verwendung des Wenn-Operators als Argument für eine [überladene Funktion](#), da der Typ des Resultats des Wenn-Operators zum Zeitpunkt der Kompilation des Programms

definiert wird. Und dieser Typ ist wie die größeren vom Typen "Ausdruck2" und "Ausdruck3" [definiert](#).

**Beispiel:**

```
void func(double d) { Print("double argument: ",d); }
void func(string s) { Print("string argument: ",s); }

bool   Expression1=true;
double Expression2=M_PI;
string Expression3="3.1415926";

void OnStart()
{
    func(Expression2);
    func(Expression3);

    func(Expression1?Expression2:Expression3); // erhalten Compiler-Warnung über exp
    func(!Expression1?Expression2:Expression3); // erhalten Compiler-Warnung über exp
; }

// Ergebnis:
// double argument: 3.141592653589793
// string argument: 3.1415926
// string argument: 3.141592653589793
// string argument: 3.1415926
```

**Sehen Sie auch**

[Initialisierung der Variablen](#), [Sichtbereich und Lebensdauer der Variablen](#), [Erzeugung und Entfernung der Objekte](#)

## Umschalteroperator switch

Vergleicht den Ausdruckswert mit Konstanten in allen Arten *case* und gibt die Steuerung des Operator, die dem Ausdruckswert entspricht. Jede Art kann von ganzzahliger Konstante, Symbolkonstante oder Konstantausdruck notiert werden. Ausdruck des Operators *switch* muss ganzzahlig sein.

```
switch (Ausdruck)
{
    case Konstante: Operatoren
    case Konstante: Operatoren
    ...
    default: Operatoren
; }
```

Operatoren, verbunden mit Vermerk *default*, werden durchgeführt, wenn keine Konstante in Operatoren *case* dem Ausdruckswert nicht gleich ist. Art *default* muss nicht unbedingt erklärt werden und nicht unbedingt am letzten sein. Wenn keine Konstante dem Ausdruckswert entspricht und die Art *default* fehlt, werden keine Handlungen durchgeführt.

Das Schlüsselwort *case* mit der Konstante sind einfach Vermerke, wenn Operatoren für die Art *case* durchgeführt werden, werden weiter Operatoren aller folgenden Arten durchgeführt, bis der Operator *break* auftritt. Das ermöglicht eine Aufeinanderfolge von Operatoren mit mehreren Varianten zu verbinden.

Konstantausdruck wird in der Etappe der Compilierung durchgeführt. Keine zwei Konstanten können in einem Operator-Umschalter gleiche Werte haben.

### Beispiele:

```
//--- erstes Beispiel
switch(x)
{
    case 'A':
        Print("CASE A");
        break;
    case 'B':
    case 'C':
        Print("CASE B or C");
        break;
    default:
        Print("NOT A, B or C");
        break;
; }

//--- zweites Beispiel
string res="";
int i=0;
switch(i)
{
    case 1:
        res=i;break;
}
```



```
    default:
        res="default";break;
    case 2:
        res=i;break;
    case 3:
        res=i;break;
    ; }
Print(res);
/*
    Ergebnis
    default
*/
```

**Sehen Sie auch**

[Initialisierung der Variablen](#), [Sichtbereich und Lebensdauer der Variablen](#), [Erzeugung und Entfernung der Objekte](#)

## Loop-Operator while

Operator **while** besteht aus einem geprüften Ausdruck und einem Operator, der erfüllt werden muss.

```
while (Ausdruck)
    Operator;
```

Wenn der Ausdruck richtig ist, wird der Operator durchgeführt, bis der Ausdruck falsch wird. Wenn der Ausdruck falsch ist, wird die Steuerung des nächsten folgenden Operators gegeben. Ausdruckswert wird vor der Operatorsdurchführung berechnet. So wenn der Ausdruck vom Anfang an falsch ist, wird der Operator nicht durchgeführt.

### Bemerkung

Wenn voraussichtlich eine große Anzahl von Iterationen in einem Zyklus behandelt wird, ist es ratsam, dass Sie die Tatsache der erzwungenen Beendigung des Programms mit der Funktion [IsStopped\(\)](#) überprüfen.

### Beispiel:

```
while (k<n && !IsStopped())
{
    y=y*x;
    k++;
; }
```

### Sehen Sie auch

[Initialisierung der Variablen](#), [Sichtbereich und Lebensdauer der Variablen](#), [Erzeugung und Entfernung der Objekte](#)

## Loop-Operator for

Operator for besteht aus drei Ausdrücken und durchgeführten Anweisung:

```
for (Ausdruck1; Ausdruck2; Ausdruck3)
    Anweisung;
```

Ausdruck1 beschreibt Zyklusinitialisierung. Ausdruck2 - Prüfung der Bedingung des Zyklussschlusses. Wenn die Bedingung richtig ist, wird der Operator des Körperzyklus **for** durchgeführt. Alles wiederholt sich, bis der Ausdruck2 falsch wird. Wenn er falsch ist, ist der Zyklus zu Ende und die Steuerung wird an den nächsten Operator weitergegeben. Ausdruck3 wird nach jeder Iteration berechnet.

Operator **for** ist der folgenden Folge der Operatoren äquivalent:

```
Ausdruck1;
while (Ausdruck2)
{
    Operator;
    Ausdruck3;
};
```

Jeder der drei oder alle drei Ausdrücke im Operator **for** können fehlen, aber man darf nicht die sie trennenden Semikolon(;) weglassen. Wenn der Ausdruck2 weggelassen wird, gilt der immer für richtig. Operator **for(;;)** stellt infiniten Zyklus dar, äquivalent des Operators **while(1)** dar. Jeder Ausdruck1 und Ausdruck3 kann aus mehreren Ausdrücken bestehen, verbunden durch den Operator Komma ','.

### Bemerkung

Wenn voraussichtlich eine große Anzahl von Iterationen in einem Zyklus behandelt wird, ist es ratsam, dass Sie die Tatsache der erzwungenen Beendigung des Programms mit der Funktion [IsStopped\(\)](#) überprüfen.

### Beispiele:

```
for (x=1; x<=7000; x++)
{
    if (IsStopped())
        break;
    Print (MathPower (x, 2));
};
//--- anderes Beispiel
for (; !IsStopped(); )
{
    Print (MathPower (x, 2));
    x++;
    if (x>10) break;
};
//--- drittes Beispiel
for (i=0, j=n-1; i<n && !IsStopped(); i++, j--) a[i]=a[j];
```

### Sehen Sie auch

Initialisierung der Variablen, Sichtbereich und Lebensdauer der Variablen, Erzeugung und Entfernung der Objekte

## Loop-Operator do while

Zyklen [for](#) und [while](#) prüfen den Abschluss am Anfang und nicht am Ende des Zyklus. Dritter Loop-Operator [do](#) - [while](#) prüft die Abschlussbedingung am Ende, nach jedem Zyklusgang. Zykluskoerper wird immer wenigstens einmal durchgeführt.

```
do
    Operator;
while (Ausdruck)
```

Zuerst wird der Operator durchgeführt, dann wird der Ausdruck berechnet. Wenn der Ausdruck richtig ist, wird der Operator wieder und wieder durchgeführt. Wenn der Ausdruck falsch wird, ist der Zyklus zu Ende.

### Bemerkung

Wenn voraussichtlich eine große Anzahl von Iterationen in einem Zyklus behandelt wird, ist es ratsam, dass Sie die Tatsache der erzwungenen Beendigung des Programms mit der Funktion [IsStopped\(\)](#) überprüfen.

### Beispiel:

```
//--- Berechnung der Aufeinanderfolge von Zahlen Fibonatschi
int counterFibonacci=15;
int i=0, first=0, second=1;
int currentFibonacciNumber;
do
{
    currentFibonacciNumber=first+second;
    Print("i = ",i,"    currentFibonacciNumber = ",currentFibonacciNumber);
    first=second;
    second=currentFibonacciNumber;
    i++; // ohne diesen Operator wird infinites Zyklus!!
};
while(i<counterFibonacci && !IsStopped());
```

### Sehen Sie auch

[Initialisierung der Variablen](#), [Sichtbereich und Lebensdauer der Variablen](#), [Erzeugung und Entfernung der Objekte](#)

## Endoperator break

Operator `break` unterbricht die Durchführung der naechsten eingebetteten Aussenoperator `switch`, `while`, `do-while` oder `for`. Steuerung wird dem Operator gegeben, die dem beendenden Operator folgt. Einer der Zwecke dieses Operators - Durchführung des Zyklus zu beenden bei der Werzuordnung einer Variable.

### Beispiel:

```
//--- Suche des ersten Nullelementes
for(i=0;i<array_size;i++)
    if(array[i]==0)
        break;
```

### Sehen Sie auch

[Initialisierung der Variablen](#), [Sichtbereich und Lebensdauer der Variablen](#), [Erzeugung und Entfernung der Objekte](#)

## Forsetzungsoperator continue

Operator [continue](#) gibt die Steuerung am Anfang der Aussenoperator des Zyklus [while](#), [do-while](#) oder [for](#), und dabei ruft Anfang der folgenden Iteration auf. Dieser Operator ist dem Operator [break](#) der Wirkung nach gegenübergestellt.

**Beispiel:**

```
//--- Summe aller Nicht-Nullobjekte
int func(int array[])
{
    int array_size=ArraySize(array);
    int sum=0;
    for(int i=0;i<array_size; i++)
    {
        if(a[i]==0) continue;
        sum+=a[i];
    }
    return(sum);
}; }
```

**Sehen Sie auch**

[Initialisierung der Variablen](#), [Sichtbereich und Lebensdauer der Variablen](#), [Erzeugung und Entfernung der Objekte](#)

## Operator der Objekterzeugung new

Operator **new** erzeugt automatisch das Objekt entsprechender Größe, ruft Objektentwurf auf und gibt [Descriptor des erzeugten Objektes](#) zurück. Beim Misserfolg gibt der Operator einen Null-Descriptor zurück, der mit der Konstante [NULL](#) verglichen werden kann.

Operator *new* kann nur für Objekte der [Klasse](#) verwendet werden, für Strukturen kann sie nicht verwendet werden. Operator *new* kann nur für Objekte komplizierter Typs verwendet werden ([Strukturen und Klassen](#)).

Operator wird nicht für Erzeugung von Objektarrays angewendet. Dafür muss man die Funktion [ArrayResize\(\)](#) benutzen.

### Beispiel:

```
//+-----+
//| Figurerzeugung |
//+-----+
void CTetrisField::NewShape()
{
    m_ypos=HORZ_BORDER;
    //--- Zufallsmaessig erzeugen wir eine der 7 moeglicher Figuren
    int nshape=rand()%7;
    switch(nshape)
    {
        case 0: m_shape=new CTetrisShape1; break;
        case 1: m_shape=new CTetrisShape2; break;
        case 2: m_shape=new CTetrisShape3; break;
        case 3: m_shape=new CTetrisShape4; break;
        case 4: m_shape=new CTetrisShape5; break;
        case 5: m_shape=new CTetrisShape6; break;
        case 6: m_shape=new CTetrisShape7; break;
    };
    //--- zeichnen wir
    if(m_shape!=NULL)
    {
        //--- Ersteinstellungen
        m_shape.SetRightBorder(WIDTH_IN_PIXELS+VERT_BORDER);
        m_shape.SetYPos(m_ypos);
        m_shape.SetXPos(VERT_BORDER+SHAPE_SIZE*8);
        //--- zeichnen wir
        m_shape.Draw();
    };
    //---
};
```

Es muss bemerkt werden, das der Objektdescriptor kein Speicheranzeiger ist.

Objekt, erzeugt vom Operator *new*, muss explizit vom Operator [delete](#) entfernt werden.

Sehen Sie auch



Initialisierung der Variablen, Sichtbereich und Lebensdauer der Variablen, Erzeugung und Entfernung der Objekte

## Operator der Objektenfernung delete

Operator `delete` entfernt das vom Operator `new` [geschaffene](#) Objekt, ruft Destruktor entsprechender Klasse und befreit den vom Objekt bestzten Speicherplatz. Als Operand wird der gueltiger Beschreiber des wirklichen Objektes verwendet. Nach der Operationsdurchführung `delete` wird der [Objektdescriptor](#) ungultig.

### Beispiel:

```
//--- entfernen wir die liegende Figur  
delete m_shape;  
m_shape=NULL;  
//--- erzeugen wir die neue Figur  
NewShape();
```

### Sehen Sie auch

[Initialisierung der Variablen](#), [Sichtbereich und Lebensdauer der Variablen](#), [Erzeugung und Entfernung der Objekte](#)

## Funktionen

Jeder Task kann in Subtasks geteilt werden, jeder von denen enwede als Kode dargestellt werden kann oder in noch kleinere Subtasks geteilt werden kann. Dieses Verfahren heisst *gestufte Konkretisierung*. Funktionen dienen zur Aufzeichnung des Programmkodes diser zu geloesten Aufgaben. Kode, der beschreibt das, was die Funktion macht, heisst Funktionsdefinition:

```
Funktionstitel
{
    Anweisungen
; }
```

Alles, was vor der ersten geschweiften Klammer steht, ist der Titel der Funktionsdefinition, und das, was innerhalb der geschweiften Klammern steht, ist *Koerper* der Funktionsdefinition. Funktionstitel besteht aus der Typbeschreibung des Rückgabewertes, dem Namen ([Identifikator](#)) und den [formellen Parametern](#). Zahl der Parameter, gegeben der Funktion, ist begrenzt und kann nicht mehr als 64 sein.

Funktion kann aus anderen Progarmmteilen so oft aufgerufen, wie es notwendig ist. Tatsaechlich bilden de Rückgabety, Funktionsidentifikator und Parametertypen *Funktionsprototyp*.

Funktionsprototyp ist die Erklaerung der Funktion, nicht aber Funktionsdefinition. Wegen der expliziten Erklaerung des Rückgabety und der Liste von Argumenttypen, sind beim Zugriff zur Funktion strikte Typenpruefung und implizite Typenumwandlungen moeglich. Besonders oft werden die Funktionserklärungen in Klassen für die Verbesserung des Kodelesbarkeit vewendet.

Funktionsdefinition muss ihrer Erklaerung voellig entsprechen. Jede erklärte Funktion muss definiert werden.

### Beispiel:

```
double                                // Typ des Rückgabewertes
linfunc (double a, double b) // Funktionsname und Parameterliste
{
    // Zusammengesetzte Anweisung
    return (a + b);           // Rückgabewert
; }
```

Anweisung [return](#) kann den Ausdruckswert, der in derselben Anweisung steht, zurückgeben. Ausdruckswert wird in den Typ des Funktionsergebnisses umgewandelt. Man kann [einfache Typen](#), [einfache Strukturen](#), [Objektanzeiger](#) rückgeben. Durch den Operator *return* können keine Felder, Klassenobjekte, Variablen der komplizierten Strukturen zurückgegeben werden.

Funktion, die den Wert nicht rückgibt, muss al Funktion mit dem Typ [void](#) beschrieben werden.

### Beispiel:

```
void errmesg(string s)
{
    Print("error: "+s);
; }
```

Parameter, gegeben der Funktion, können vorgegebene Werte haben, die von Konstanten des entsprechenden Typs vorgegeben werden.

**Beispiel:**

```
int somefunc(double a,
             double d=0.0001,
             int n=5,
             bool b=true,
             string s="passed string")
{
    Print("Pflichtparameter a = ",a);
    Print("Folgende Parameter übertragen werden: d = ",d," n = ",n," b = ",b," s = ",s)
    return(0);
};
```

Wenn einem Parameter Default-Wert vorgegeben wird, müssen alle folgenden Parameter auch Default-Werte haben.

**Beispiel der nicht korrekten Erklärung :**

```
int somefunc(double a,
             double d=0.0001,    // Default-Wert ist erklärt 0.0001
             int n,              // Default-Wert nicht angegeben !
             bool b,            // Default-Wert nicht angegeben !
             string s="passed string")
{
};
```

Sehen Sie auch

[Überladen](#), [Virtuelle Funktionen](#), [Polymorphismus](#)

## Funktionsaufruf

Wenn einer Name, der früher nicht beschrieben wurde, im Ausdruck erscheint, und linke runde Klammer ihm folgt, gilt der nach dem Kontext der Name einer Funktion.

```
Funktionsname (x1, x2, ..., xn)
```

Argumente ([formelle Parameter](#)) werden nach der Bedeutung übertragen, d.h. jede Bedeutung  $x_1, \dots, x_n$  wird berechnet und der Wert wird der Funktion übertragen. Reihenfolge der Ausdruckberechnungen und Reihenfolge des Wertladens ist nicht garantiert. Während der Durchführung wird die Prüfung der Zahl und des Typs von Argumenten durchgeführt, die der Funktion übertragen wurden. Dieses Verfahren des Funktionszugriffs heisst Wertaufwurf.

Funktionsaufruf ist Ausdruck dessen Wert der Wert ist, der von der Funktion zurückgegeben wird. Der beschriebene Funktionstyp muss dem Typ des Rückgabewertes entsprechen. Funktion kann erklärt oder beschrieben werden in jeder Programmstelle auf dem [globalen Niveau](#), d.h. ausser anderer Funktionen. Funktion kann nicht innerhalb der anderen Funktion erklärt oder beschrieben werden.

### Beispiele:

```
int start()
{
    double some_array[4]={0.3, 1.4, 2.5, 3.6};
    double a=linfunc(some_array, 10.5, 8);
    //...
};

double linfunc(double x[], double a, double b)
{
    return (a*x[0] + b);
};
```

Beim Funktionsaufruf mit Default-Werten kann die Liste der übertragenen Parameter vor dem ersten Parameter mit Default-Wert nicht begrenzt werden.

### Beispiele:

```
void somefunc(double init,
              double sec=0.0001, //Default-Werte sind bestimmt
              int level=10);

//...

somefunc(); // falscher Aufruf. der erste Pflichtparameter muss sein
somefunc(3.14); // richtiger Aufruf
somefunc(3.14,0.0002); // richtiger Aufruf
somefunc(3.14,0.0002,10); // richtiger Aufruf
```

Beim Funktionsaufruf darf man nicht Default-Werte auslassen, auch wenn sie Default-Werte haben:

```
somefunc(3.14, , 10); // falscher Aufruf -> zweiter Parameter ist ausgelassen
```

### Sehen Sie auch

[überladen](#), [Virtuelle Funktionen](#), [Polymorphismus](#)

## Parameterübertragung

Es gibt zwei Verfahren, durch die Computersprache das Argument dem Subprogramm (der Funktion) übertragen kann. Das erste Verfahren - Parameterübertragung nach der Bedeutung. Dieses Verfahren kopiert den Wert von Argument in den formellen Funktionsparameter. Darum haben verschiedene Änderungen dieses Parameters keinen Einfluss auf das entsprechende Aufrufsargument.

```
//+-----+
//| Parameterübergabe nach dem Wert |
//+-----+
double FirstMethod(int i,int j)
{
    double res;
//---
    i*=2;
    j/=2;
    res=i+j;
//---
    return(res);
;}

//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//---
    int a=14,b=8;
    Print("a und b vor dem Aufruf:",a," ",b);
    double d=FirstMethod(a,b);
    Print("a und b nach dem Aufruf:",a," ",b);
;}

//--- Ergebnis der Scriptdurchführung
// a und b vor dem Aufruf: 14 8
// a und b nach dem Aufruf: 14 8
```

Zweites Verfahren - Parameterübertragung durch Referenz. In diesem Fall wird der Verweis auf Parameter (nicht aber auf seine Bedeutung\_ dem Funktionsparameter übertragen. Innerhalb der Funktion wird der Verweis verwendet, um zum tatsächlichen Parameter, angegebenen im Aufruf, zu greifen. Das bedeutet, dass Parameteränderungen das Argument beeinflussen werden, das für Funktionsaufruf verwendet wurde.

```
//+-----+
//| Parameterübergabe nach Verweis |
//+-----+
double SecondMethod(int &i,int &j)
{
    double res;
//---
    i*=2;
    j/=2;
```

```

    res=i+j;
//---
    return(res);
;}
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//---
    int a=14,b=8;
    Print("a und b vor dem Aufruf:",a," ",b);
    double d=SecondMethod(a,b);
    Print("a und b nach dem Aufruf:",a," ",b);
;}
//+-----+
//--- Ergebnis der Scriptdurchführung
//  a und b vor dem Aufruf: 14 8
//  a und b nach dem Aufruf: 28 4

```

MQL5 verwendet beide Verfahren, mit Ausnahme von: Felder, Variablen des Typs Strukturen und Klassenobjekte werden immer durch Verweis übertragen. Um Änderungen der tatsächlichen Parameter auszuschliessen (Argumente, übertragen beim Funktionsaufruf), muss man Zugriffsspezifikator const verwenden. Beim Versuch, Inhalt der Variable zu verändern, die mit dem Spezifikator `const` erklärt wurde, generiert der Compiler einen Fehler.

## Bemerkung

Es muss behalten werden, dass Parameter der Funktion umgekehrt übertragen werden, d.h. am Anfang wird der letzte Parameter berechnet und übertragen, dann vorletzte usw. Am letzten wird der Parameter berechnet und übertragen, der am ersten nach den Klammern steht.

### Beispiel:

```

void OnStart()
{
//---
    int a[]={0,1,2};
    int i=0;

    func(a[i],a[i++],"Erste Aufruf(i="+string(i)+"");
    func(a[i++],a[i],"Zweiter Aufruf(i="+string(i)+"");
// Ergebnis:
// Erste Aufruf(i=0) : par1 = 1    par2 = 0
// Zweiter Aufruf(i=1) : par1 = 1    par2 = 1

;}
//+-----+
//| |
//+-----+

```

```
void func(int par1,int par2,string comment)
{
    Print(comment,": par1 =",par1,"    par2 =",par2);
};
```

Beim ersten Aufruf im angeführten Beispiel nimmt die Variable *i* am Anfang an der Zahlenkonkatenanz teil.

```
"Erster Aufruf(i="+string(i)+"") "
```

Dabei verändert sich ihr Wert nicht. Dann beteiligt sich die Variable *i* an der Berechnung des Elementes vom Feld ***a[i++]***, d.h. nach der Entnahme des *i*-Elementes des Feldes wird die Variable *i* inkrementiert. Und erst danach wird der erste Parameter mit dem veränderten Wert der Variable *i* berechnet.

Im zweiten Aufruf bei der Berechnung aller drei Parameter wird derselbe Wert *i* verwendet, der beim ersten Funktionsaufruf verändert wurde, und erst nach der Berechnung des ersten Parameters verändert sich die Variable *i* wieder.

Sehen Sie auch

[Sichtbereich und Lebensdauer der Variablen](#), [überladen](#), [Virtuelle Funktionen](#), [Polymorphismus](#)



## Überladung der Funktionen

Gewöhnlich versucht man im Namen der Funktion ihre Hauptbedeutung darzustellen. In der Regel haben lesbare Programme verschiedenartige und gut passende [Identifikatoren](#). Manchmal werden verschiedene Funktionen für dieselbe Zwecke verwendet. Betrachten wir zB die Funktion, die den Durchschnittwert des Feldes von Zahlen mit Doppelgenauigkeit berechnet und dieselbe Funktion, die aber mit Feldern der Ganzzahlen operiert. Sowie Die erste, als auch die zweite benenne wir AverageFromArray:

```
//+-----+
//| Berechnung des Durchschnittwertes für Feld der Art double |
//+-----+
double AverageFromArray(const double & array[],int size)
{
    if(size<=0) return 0.0;
    double sum=0.0;
    double aver;
//---
    for(int i=0;i<size;i++)
    {
        sum+=array[i];    // Addition für double
    }
    aver=sum/size;        // Dividieren wir einfach die Summe durch die Zahl
//---
    Print("Berechnung des Durchschnittwertes für Feld der Art double");
    return aver;
};

//+-----+
//| Berechnung des Durchschnittwertes für Feld der Art int |
//+-----+
double AverageFromArray(const int & array[],int size)
{
    if(size<=0) return 0.0;
    double aver=0.0;
    int sum=0;
//---
    for(int i=0;i<size;i++)
    {
        sum+=array[i];    // Addition für int
    }
    aver=(double)sum/size;// reduzieren wir die Summe zum Typ double und dividieren
//---
    Print("Berechnung des Durchschnittwertes für Feld der Art int");
    return aver;
};
```

Jede Funktion hat Nachrichtenausgabe mittels der Funktion [Print\(\)](#);

```
Print("Berechnung des Durchschnittwertes für Feld der Art int");
```

Compiler wählt die notwendige Funktion der Argumententypen und deren Zahl entsprechend. Regel, nach der diese Wahl durchgeführt wird, heisst *Algorithmus der Signaturübereinstimmung*. Unter Signatur wird die Typenliste verstanden, die in der Funktionserklärung verwendet wird.

**Beispiel:**

```
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//---
    int    a[5]={1,2,3,4,5};
    double b[5]={1.1,2.2,3.3,4.4,5.5};
    double int_aver=AverageFromArray(a,5);
    double double_aver=AverageFromArray(b,5);
    Print("int_aver=",int_aver,"    double_aver=",double_aver);
; }
//--- Ergebnis der Scriptarbeit
// Berechnung des Durchschnittswertes für Feld der Art int
// Berechnung des Durchschnittswertes für Feld der Art double
// int_aver= 3.00000000    double_aver= 3.30000000
```

Überladung ist Zuordnung der mehreren Werte einer Funktion. Die Auswahl des konkreten Wertes hängt von Argumententypen ab, die die Funktion erhalten hat. Bestimmte Funktion wird abhängig davon ausgewählt, ob die Liste der Argumente beim Funktionsaufruf der Liste der Parameter in Funktionserklärung entspricht oder nicht.

Wenn überladene Funktion aufgerufen wird, muss der Compiler Algorithmus für Auswahl notwendiger Funktion haben. Algorithmus, der diese Wahl trifft, hängt von dem [Umwandlungstyp](#) ab. Die beste Übereinstimmung muss einzigartig sein. Sie muss am besten für wenigstens ein Argument sein und so gut, wie andere Übereinstimmungen für alle anderen Argumente sein.

Unten ist Algorithmus der Übereinstimmung für jedes Argument angegeben.

## Algorithmus der Wahl von überladener Funktion

1. strikte Übereinstimmung verwenden (wenn es möglich ist).
2. Standarderhöhung des Typs probieren.
3. Standardumwandlung des Typs probieren.

Standarderhöhung des Typs ist besser, als andere Standardumwandlungen. Erhöhung ist Umwandlung von **float** in **double**, aber auch von **bool**, **char**, **short** oder **enum** in **int**. Außerdem gehört zu Standardumwandlungen Umwandlungen der Felder von ähnlichen [Ganzzahligen Typen](#). Ganzzahlige Typen sind: **bool**, **char**, **uchar**, denn alle drei Typen sind ganzzahlige 1-Byte Typen; 2-Byte ganzzahlige Typen **short** und **ushort**; 4-Byte ganzzahlige Typen **int**, **uint** und **color**; **long**, **ulong** und **datetime**.

Bestimmt ist strikte Übereinstimmung am besten. Für diese Übereinstimmung können [Typenreduzierungen](#) verwendet werden. Compiler wird nicht die doppeldeutige Situation schaffen. Darum soll man nicht auf feine Unterschiede unter Typen und explizite Umwandlungen Wert legen, die gestaute Funktion unklar machen.

Wenn Sie zweifeln, verwenden Sie explizite Umwandlungen, um strikte Übereinstimmung zu gewährleisten.

Beispiele überladener Funktionen in MQL5 können Sie am Beispiel der Funktionen [ArrayInitialize\(\)](#) beobachten.

Überladungsregeln der Funktion ist für [Überladung der Klassenmethoden](#) anwendbar.

Überladen der Systemfunktionen ist zulässig, dabei muss man verfolgen, dass Compiler die nötige Funktion genau wählen kann. ZB können wir Systemfunktion [fmax\(\)](#) mittels der drei verschiedenen Verfahren stauen, aber nur zwei von ihnen werden korrekt sein.

#### Beispiel:

```
// Stauung ist zulässig - unterscheidet sich nach der Zahl der Parameter
double fmax(double a, double b, double c);

// Stauung mit Fehler
// Parameterzahl ist verschieden, aber der letzte hat Default-Wert
// das führt zum Kapseln der Systemfunktion beim Aufruf, was unzulässig ist
double fmax(double a, double b, double c=DBL_MIN);

// normale Überladung nach der Parameterzahl
int fmax(int a, int b);
```

#### Sehen Sie auch

[Überladen](#), [Virtuelle Funktionen](#), [Polymorphismus](#)

## Überladen von Operatoren

Zur leichten Lesen und Schreiben von Code ist Überladen einiger Operationen erlaubt. Ein Überladenoperator wird mit dem Schlüsselwort **Betreiber** geschrieben. Die folgenden Operatoren können überladen werden:

- binäre +, -, /, \*, %, <<, >>, ==, !=, <, >, <=, >=, +=, -=, /=, \*=, %=, &=, |=, ^=, <<=, >>=, &&, ||, &, |, ^;
- unäre +, -, ++, --, !, ~;
- Zuweisungsoperator =;
- Indizierungsoperator [].

Überladen von Operatoren ermöglicht den Einsatz des Betriebssystems Notation (geschrieben in der Form von einfachen Ausdrücken) zu komplexen Objekte - Strukturen und Klassen. Schreiben von Ausdrücken mit überladenen Operationen vereinfacht den Blick auf den Quellcode, weil eine komplexere Implementierung verborgen ist.

Betrachten wir zum Beispiel komplexe Zahlen, die aus Real- und Imaginärteil bestehen. Sie sind weit verbreitet in der Mathematik weit verbreitet. Die MQL5 Sprache hat keinen Datentyp für Darstellung der komplexen Zahlen, aber es ist möglich, einen neuen Datentyp in Form einer Struktur oder Klasse zu erstellen. Wir deklarieren die Struktur complex und definieren vier Methoden in dieser Struktur, die die vier arithmetischen Operationen implementieren:

```
//+-----+
//| Struktur für Operationen mit komplexen Zahlen |
//+-----+
struct complex
{
    double      re; // Realteil
    double      im; // Imaginärteil
    //--- Konstruktoren
        complex():re(0.0),im(0.0) { }
        complex(const double r):re(r),im(0.0) { }
        complex(const double r,const double i):re(r),im(i) { }
        complex(const complex &o):re(o.re),im(o.im) { }

    //--- Rechenoperationen
    complex      Add(const complex &l,const complex &r) const; // Addition
    complex      Sub(const complex &l,const complex &r) const; // Subtraktion
    complex      Mul(const complex &l,const complex &r) const; // Multiplikation
    complex      Div(const complex &l,const complex &r) const; // Division
};
```

Jetzt können wir Variablen, die komplexen Zahlen vertreten, in unserem Code deklarieren und mit ihnen arbeiten.

Zum Beispiel:

```
void OnStart()
{
    //--- Deklarieren und initialisieren Variablen eines komplexen Typs
    complex a(2,4),b(-4,-2);
```

```

    PrintFormat("a=%.2f+i*%.2f,    b=%.2f+i*%.2f",a.re,a.im,b.re,b.im);
//--- Addition zweier Zahlen
    complex z;
    z=a.Add(a,b);
    PrintFormat("a+b=%.2f+i*%.2f",z.re,z.im);
//--- Multiplikation zweier Zahlen
    z=a.Mul(a,b);
    PrintFormat("a*b=%.2f+i*%.2f",z.re,z.im);
//--- Division zweier Zahlen
    z=a.Div(a,b);
    PrintFormat("a/b=%.2f+i*%.2f",z.re,z.im);
//---
; }

```

Aber es wäre bequemer für die üblichen arithmetischen Operationen mit komplexen Zahlen die üblichen Operatoren "+", "-", "\*" und "/" zu verwenden.

Das Schlüsselwort **operator** wird verwendet um die Memberfunktion, die eine Typkonvertierung führt, zu definieren. Unäre und binäre Operationen für Variable-Klassenobjekte können als nicht-statische Memberfunktionen überladen werden. Sie implizit auf das Objekt der Klasse wirken.

Die meisten binären Operatoren können wie normale Funktionen, die eine oder beide Argumente als Variable der Klasse oder als ein Zeiger auf ein Objekt dieser Klasse nehmen, überladen werden. Für unseren Typ complex wird Überladen im Deklarieren wie folgt aussehen:

```

//--- Operatoren
complex operator+(const complex &r) const { return(Add(this,r)); }
complex operator-(const complex &r) const { return(Sub(this,r)); }
complex operator*(const complex &r) const { return(Mul(this,r)); }
complex operator/(const complex &r) const { return(Div(this,r)); }

```

Ein komplettes Beispiel des Skripts:

```

//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//--- deklarieren und initialisieren Variablen eines komplexen Typs
complex a(2,4),b(-4,-2);
PrintFormat("a=%.2f+i*%.2f,    b=%.2f+i*%.2f",a.re,a.im,b.re,b.im);
//a.re=5;
//a.im=1;
//b.re=-1;
//b.im=-5;
//--- addieren zwei Zahlen
complex z=a+b;
PrintFormat("a+b=%.2f+i*%.2f",z.re,z.im);
//--- Multiplikation zweier Zahlen

z=a*b;
PrintFormat("a*b=%.2f+i*%.2f",z.re,z.im);
//--- Division zweier Zahlen
z=a/b;
PrintFormat("a/b=%.2f+i*%.2f",z.re,z.im);
//---
}
//+-----+
//| eine Struktur für Operationen mit komplexen Zahlen |
//+-----+
struct complex
{
double          re; // Realteil
double          im; // Imaginärteil
//--- Konstruktoren
complex():re(0.0),im(0.0) { }
complex(const double r):re(r),im(0.0) { }
complex(const double r,const double i):re(r),im(i) { }
complex(const complex &o):re(o.re),im(o.im) { }

//--- arithmetische Operationen
complex          Add(const complex &l,const complex &r) const; // Addition
complex          Sub(const complex &l,const complex &r) const; // Subtraktion
complex          Mul(const complex &l,const complex &r) const; // Multiplikation
complex          Div(const complex &l,const complex &r) const; // Division
//--- binäre Operatoren
complex operator+(const complex &r) const { return(Add(this,r)); }
complex operator-(const complex &r) const { return(Sub(this,r)); }
complex operator*(const complex &r) const { return(Mul(this,r)); }
complex operator/(const complex &r) const { return(Div(this,r)); }
};
//+-----+
//| Addition |
//+-----+
complex complex::Add(const complex &l,const complex &r) const
{
complex res;
//---
res.re=l.re+r.re;
res.im=l.im+r.im;
//--- Ergebnis
return res;
}
//+-----+
//| Subtraktion |

```

```

//+-----+
complex complex::Sub(const complex &l,const complex &r) const
{
    complex res;
//---
    res.re=l.re-r.re;
    res.im=l.im-r.im;
//--- Ergebnis
    return res;
}
//+-----+
//| Multiplikation |
//+-----+
complex complex::Mul(const complex &l,const complex &r) const
{
    complex res;
//---
    res.re=l.re*r.re-l.im*r.im;
    res.im=l.re*r.im+l.im*r.re;
//--- Ergebnis
    return res;
}
//+-----+
//| Division |
//+-----+
complex complex::Div(const complex &l,const complex &r) const
{
//--- leere komplexe Zahl
    complex res(EMPTY_VALUE,EMPTY_VALUE);
//--- prüfen zu Null
    if(r.re==0 && r.im==0)
    {
        Print(__FUNCTION__+": number is zero");
        return(res);
    }
//--- Hilfsvariable
    double e;
    double f;
//--- Wahl der Rechenvariante
    if(MathAbs(r.im)<MathAbs(r.re))
    {
        e = r.im/r.re;
        f = r.re+r.im*e;
        res.re=(l.re+l.im*e)/f;
        res.im=(l.im-l.re*e)/f;
    }
    else
    {
        e = r.re/r.im;
        f = r.im+r.re*e;
        res.re=(l.im+l.re*e)/f;
        res.im=(-l.re+l.im*e)/f;
    }
//--- Ergebnis
    return res;
}

```

Die meisten unären Operationen für Klassen können als die üblichen Funktionen, die ein einziges Argument-Klassenobjekt oder einen Zeiger auf es nehmen, überladen werden. Fügen wir eine

Überladen unäre Operationen "-" und "!".

```
//+-----+
//| Eine Struktur für Operationen mit komplexen Zahlen |
//+-----+
struct complex
{
    double      re;      // Realteil
    double      im;      // Imaginärteil
    ...
    //--- unäre Operatoren
    complex operator-() const; // unäres Minus
    bool      operator!() const; // Negation
};
...
//+-----+
//| Überladen von Operator "unäres Minus" |
//+-----+
complex complex::operator-() const
{
    complex res;
    //---
    res.re=-re;
    res.im=-im;
    //--- Ergebnis
    return res;
}
//+-----+
//| Überladen von Operator "logische Negation" |
//+-----+
bool complex::operator!() const
{
    //--- Sind die realen und imaginären Teile der komplexen Zahl gleich Null?
    return (re!=0 && im!=0);
}
```

Jetzt können wir den Wert einer komplexen Zahl durch Null prüfen und einen negativen Wert erhalten:

```
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    //--- deklarieren und initialisieren Variablen eines komplexen Typs
    complex a(2,4),b(-4,-2);
    PrintFormat("a=%.2f+i*%.2f, b=%.2f+i*%.2f",a.re,a.im,b.re,b.im);
    //--- Division zweier Zahlen z=a/b;
    complex z=a/b;
    PrintFormat("a/b=%.2f+i*%.2f",z.re,z.im);
    //--- Eine komplexe Zahl wird standardmäßig Null gleich (im Standardkonstruktor re==0
    complex zero;
    Print("!zero=",!zero);
    //--- einen negativen Wert anweisen
    zero=-z;
    PrintFormat("z=%.2f+i*%.2f, zero=%.2f+i*%.2f",z.re,z.im, zero.re,zero.im);
    PrintFormat("-zero=%.2f+i*%.2f",-zero.re,-zero.im);
    //--- prüfen wir zu Null noch einmal
    Print("!zero=",!zero);
    //---
}
```



```
}
```

Bitte beachten Sie, dass wir nicht in diesem Fall den Zuweisungsoperator "=" überladen haben, wie [Strukturen von einfachen Typen](#) können eins in andere direkt kopiert werden. So können wir nun den Code für Berechnungen mit komplexen Zahlen in der üblichen Weise schreiben.

Überladen von Index-Operator ermöglicht Erhalten von im Objekt eingeschlossenen Arrays auf einfache und gewohnte Art, und dies trägt auch zu einem besseren Verständnis und Lesbarkeit des Quellcodes. Zum Beispiel müssen wir den Zugang zu einem Symbol im String an der angegebenen Position vorsehen. Ein String in der Sprache MQL5 ist ein eigener Typ [string](#), der nicht ein Array von Symbolen ist, sondern mit Hilfe einer überladenen Index-Operation in der generierten Klasse CString können wir eine einfache und transparente Arbeit stellen:

```
//+-----+
//| Klasse, um Symbole im String als Array in Symbole zu zugreifen |
//+-----+
class CString
{
    string          m_string;

public:
    CString(string str=NULL):m_string(str) { }
    ushort operator[] (int x) { return(StringGetCharacter(m_string,x)); }
};
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    //--- Array für Erhalten der Symbole aus einem String
    int      x[]={ 19,4,18,19,27,14,15,4,17,0,19,14,17,27,26,28,27,5,14,
                    17,27,2,11,0,18,18,27,29,30,19,17,8,13,6 };
    CString str("abcdefghijklmnopqrstuvwxy[ ]CS");
    string  res;
    //--- Bilden einen Satz mit Symbolen aus der Variable str
    for(int i=0,n=ArraySize(x);i<n;i++)
    {
        res+=ShortToString(str[x[i]]);
    }
    //--- Ausgabe von Ergebnisse
    Print(res);
}
```

Ein weiteres Beispiel für ein Überladen der Index-Operation ist Arbeit mit Matrizen. Die Matrix ist ein zweidimensionales dynamisches Array. Größe des Arrays sind nicht im Voraus definiert. So kann man nicht ein Array von Form `array[][]` ohne Angabe der Größe der zweiten Dimension deklarieren, und dann dieses Array als Parameter übergeben. Eine mögliche Lösung ist eine spezielle Klasse CMatrix, die ein Array von Objekten der Klasse CRow enthält.

```

//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//--- Operationen der Addition und Multiplikation von Matrizen
    CMatrix A(3),B(3),C();
//--- bereiten die Arrays für Strings
    double a1[3]={1,2,3}, a2[3]={2,3,1}, a3[3]={3,1,2};
    double b1[3]={3,2,1}, b2[3]={1,3,2}, b3[3]={2,1,3};
//--- füllen die Matrizen
    A[0]=a1; A[1]=a2; A[2]=a3;
    B[0]=b1; B[1]=b2; B[2]=b3;
//--- Ausgabe von Matrizen in "Experts" Journal
    Print("---- Elemente der Matrix A");
    Print(A.String());
    Print("---- Elemente der Matrix B");
    Print(B.String());

//--- Multiplikation von Matrizen
    Print("---- Multiplikation von Matrizen A und B");
    C=A+B;
//--- Ausgabe der formatierten String-Darstellung
    Print(C.String());

//---Multiplikation von Matrizen
    Print("---- Produkt von Matrizen A und B");
    C=A*B;
    Print(C.String());

//--- jetzt zeigen wir wie Werte im Stil des dynamischen Arrays matrix[i][j] zu erhalten
    Print("die Werte der Matrix C elementweise ausgeben");
//--- Durchschleifen der Strings der Matrix - Objekte CRow
    for(int i=0;i<3;i++)
    {
        string com="| ";
        //--- bilden Strings aus der Matrix für den Wert
        for(int j=0;j<3;j++)
        {
            //--- erhalten ein Element der Matrix aus Nummer von Zeile und Spalte
            double element=C[i][j]; // [i] - Zugriff zu CRow im Array m_rows[] ,
                                     // [j] - Überladener Index-Operator in CRow
            com=com+StringFormat("a(%d,%d)=%G ; ",i,j,element);
        }
        com+="| ";
        //--- Die Werte von der Zeile ausgeben
        Print(com);
    }
}
//+-----+
//| Klasse "Zeile" |
//+-----+
class CRow
{
private:
    double          m_array[];
public:
    //--- Konstruktoren und Destruktor
    CRow(void)       { ArrayResize(m_array,0); }
    CRow(const CRow &r) { this=r; }
    CRow(const double &array[]);
}

```

```

~CRow(void) {};

//--- Anzahl der Elementen in der Zeile
int      Size(void) const { return(ArraySize(m_array));}
//--- Gibt den String mit Werte zurück
string   String(void) const;
//--- Index-Operator
double   operator[](int i) const { return(m_array[i]); }
//--- Zuweisungsoperatoren
void     operator=(const double &array[]); // Array
void     operator=(const CRow & r);        // ein anderes Objekt CRow
double   operator*(const CRow &o);         // Objekt CRow für Multiplikation
};

//+-----+
//| Konstruktor zur Initialisierung einer Zeile mit einem Array |
//+-----+
void CRow::CRow(const double &array[])
{
    int size=ArraySize(array);
    //--- Wenn das Array nicht leer ist
    if(size>0)
    {
        ArrayResize(m_array,size);
        //--- Füllen mit Arrays
        for(int i=0;i<size;i++)
            m_array[i]=array[i];
    }
    //---
}

//+-----+
//| Zuweisungsoperation für das Array |
//+-----+
void CRow::operator=(const double &array[])
{
    int size=ArraySize(array);
    if(size==0) return;
    //--- füllen das Array mit Werten
    ArrayResize(m_array,size);
    for(int i=0;i<size;i++) m_array[i]=array[i];
    //---
}

//+-----+
//| Zuweisungsoperation für CRow |
//+-----+
void CRow::operator=(const CRow &r)
{
    int size=r.Size();
    if(size==0) return;
    //--- füllen das Array mit Werten
    ArrayResize(m_array,size);
    for(int i=0;i<size;i++) m_array[i]=r[i];
    //---
}

//+-----+
//| Operator der Multiplikation auf einer anderen Zeile |
//+-----+
double CRow::operator*(const CRow &o)
{
    double res=0;
    //--- Prüfung
    int size=Size();
    if(size!=o.Size() || size==0)

```

```

    {
        Print(__FUNCSIG__, ": Fehler bei der Multiplikation zweier Matrizen, Die Größe si
        return(res);
    }
//--- multiplizieren die Arrays elementweise und addieren die Produkte
    for(int i=0;i<size;i++)
        res+=m_array[i]*o[i];
//--- Ereignis
    return(res);
}
//+-----+
//| Gibt die formatierte String-Darstellung zurück |
//+-----+
string CRow::String(void) const
{
    string out="";
//--- wenn die Größe des Arrays mehr als Null ist
    int size=ArraySize(m_array);
//--- arbeiten nur mir Nicht-Null Anzahl der Elemente im Array
    if(size>0)
    {
        out="{ ";
        for(int i=0;i<size;i++)
        {
            //--- sammeln Werte in einen String
            out+=StringFormat(" %G;",m_array[i]);
        }
        out+=" }";
    }
//--- Ergebnis
    return(out);
}
//+-----+
//| Klasse "Matrix" |
//+-----+
class CMatrix
{
private:
    CRow          m_rows[];

public:
    //--- Konstruktoren und Destruktor
        CMatrix(void);
        CMatrix(int rows) { ArrayResize(m_rows,rows); }
        ~CMatrix(void){};

    //--- Erhalten von Größen der Matrix
    int      Rows()      const { return(ArraySize(m_rows)); }
    int      Cols()      const { return(Rows()>0? m_rows[0].Size():0); }
//--- gibt den Wert der Spalte als String CRow zurück
    CRow     GetColumnAsRow(const int col_index) const;
//--- gibt einen String mit den Werten der Matrix zurück
    string   String(void) const;
//--- Index-Operator gibt eine Zeile auf ihrer Nummer zurück
    CRow *operator[](int i) const { return(GetPointer(m_rows[i])); }
//--- Additionoperator
    CMatrix  operator+(const CMatrix &m);
//--- Multiplikationsoperator
    CMatrix  operator*(const CMatrix &m);
//--- Zuweisungsoperator
    CMatrix  *operator=(const CMatrix &m);
};

```

```

//+-----+
//| Standardkonstruktor, erstellt Array von Strings mit Null-Größe |
//+-----+
CMatrix::CMatrix(void)
{
//--- Null Zeilen in der Matrix
    ArrayResize(m_rows,0);
//---
}
//+-----+
//| Gibt den Wert der Spalte als String CRow zurück |
//+-----+
CRow CMatrix::GetColumnAsRow(const int col_index) const
{
//--- Variable, um die Werte aus Spalte zu erhalten
    CRow row();
//--- Anzahl der Zeile in der Matrix
    int rows=Rows();
//--- Wenn die Anzahl der Zeilen größer als Null ist, führen die Operation
    if(rows>0)
    {
        //--- Array für Erhalten von Werte der Spalte mit Index col_index
        double array[];
        ArrayResize(array,rows);
        //--- füllen das Array
        for(int i=0;i<rows;i++)
        {
            //--- Prüfung der Spaltennummer für Zeile i, ob es über die Grenzen des Array
            if(col_index>=this[i].Size())
            {
                Print(__FUNCSIG__,": Fehler! Nummer der Spalte ",col_index,"> Größe der Zeile");
                break; // row bleibt ein nicht initialisiertes Objekt
            }
            array[i]=this[i][col_index];
        }
        //--- erstellen wir eine Zeile CRow basierend auf den Werten des Arrays
        row=array;
    }
//--- Ergebnis
    return(row);
}
//+-----+
//| Multiplikation zweier Matrizen |
//+-----+
CMatrix CMatrix::operator+(const CMatrix &m)
{
//--- die Anzahl der Zeilen und Spalten in der übergebene Matrix
    int cols=m.Cols();
    int rows=m.Rows();
//--- Matrix, um das Ergebnis der Addition zu erhalten
    CMatrix res(rows);
//--- Dimensionen der Matrix müssen identisch sein
    if(cols!=m.Cols() || rows!=m.Rows())
    {
        //--- kann nicht addieren
        Print(__FUNCSIG__,": Fehler bei der Addition zweier Matrizen, die Größe sind nicht identisch");
        return(res);
    }
//--- Hilfsarray
    double arr[];
    ArrayResize(arr,cols);

```

```

//--- Iteration durch die Zeilen für Addition
for(int i=0;i<rows;i++)
{
    //--- wir schreiben das Ergebnis der Addition von Zeilen der Matrix in ein Array
    for(int k=0;k<cols;k++)
    {
        arr[k]=this[i][k]+m[i][k];
    }
    //--- stellen das Array in der Zeile der Matrix
    res[i]=arr;
}
//--- Ergebnis der Addition zweier Matrizes zurückgeben
return(res);
}
//+-----+
//| Multiplikation zweier Matrizen |
//+-----+
CMatrix CMatrix::operator*(const CMatrix &m)
{
    //--- Anzahl der Spalten der ersten Matrix Anzahl der Zeilen in der übergebenen Matrix
    int cols1=Cols();
    int rows2=m.Rows();
    int rows1=Rows();
    int cols2=m.Cols();
    //--- Matrix, um das Ergebnis der Addition zu erhalten
    CMatrix res(rows1);
    //--- Matrizes sollten abgestimmt werden
    if(cols1!=rows2)
    {
        //--- Kann nicht multiplizieren
        Print(__FUNCSIG__,": Fehler bei der Multiplikation zweier Matrizen, ist das Form
            "- Anzahl der Spalten in den ersten Faktor sollte gleich der Anzahl der Ze
        return(res);
    }
    //--- Hilfsarray
    double arr[];
    ArrayResize(arr,cols1);
    //--- füllen Zeilen in der Matrix von Multiplikation
    for(int i=0;i<rows1;i++)// iterieren durch die Zeilen
    {
        //--- Array-Empfänger auf Null setzen
        ArrayInitialize(arr,0);
        //--- iterieren Elemente in der Zeile
        for(int k=0;k<cols1;k++)
        {
            //--- Nehmen Werte der Spalte k der Matrix m als ein String CRow
            CRow column=m.GetColumnAsRow(k);
            //--- multiplizieren die beiden Zeilen und schreibt das Ergebnis der Skalarmu
            arr[k]=this[i]*column;
        }
        //--- setzen Array arr[] in der Zeile i der Matrix
        res[i]=arr;
    }
    //--- Produkt zweier Matrizen zurückgeben
    return(res);
}
//+-----+
//| Zuweisungsoperation |
//+-----+
CMatrix *CMatrix::operator=(const CMatrix &m)
{

```

```
//--- finden und definieren die Anzahl der Zeilen
int rows=m.Rows();
ArrayResize(m_rows,rows);
//--- füllen unsere Strings mit den Werten der Zeilen der übergebenen Matrix
for(int i=0;i<rows;i++) this[i]=m[i];
//---
return(GetPointer(this));
}
//+-----+
//| Eine String-Darstellung der Matrix |
//+-----+
string CMatrix::String(void) const
{
    string out="";
    int rows=Rows();
    //--- bilden Zeile für Zeile
    for(int i=0;i<rows;i++)
    {
        out=out+this[i].String()+"\r\n";
    }
    //--- Ergebnis
    return(out);
}
```

#### Sehen Sie auch

[Überladen, Arithmetische Operationen](#), [Überladen von Funktionen](#), [Prioritäten und die Reihenfolge der Operationen](#)

## Beschreibung der Aussenfunktionen

Typ Aussenfunktionen, bestimmt in anderem Mode, muss explizit beschrieben werden. Fehlen solcher Beschreibung kann zu Fehlern bei der Compilierung, Verknuepfung oder Programmverarbeitung führen. Bei der Beschreibung des Aussenobjektes verwenden Sie Schluesselwort `extern` mit dem Modulandeutung.

### Beispiele:

```
#import "user32.dll"
    int      MessageBoxW(int hWnd ,string szText,string szCaption,int nType);
    int      SendMessageW(int hWnd,int Msg,int wParam,int lParam);
#import "lib.ex5"
    double   round(double value);
#import
```

Mittels des Imports kann man sehr einfach Funktionen beschreiben, aufgerufen von externen DLL oder kompilierten EX5 Bibliotheken. Bibliotheken EX5 sind kompilierte ex5-Dateien, die die Eigenschaft [library](#) haben. Aus EX5 Bibliotheken können nur Funktionen importiert werden, die mit dem [Modifikator export](#) beschrieben werden.

Bei der gemeinsame Nutzung der DLL und EX5-Bibliotheken erinnern Sie daran, dass die Bibliotheken verschiedene Namen (unabhängig von Platzierung ihrer Verzeichnisse) haben müssen. Alle importierten Funktionen erhalten Scope "Dateiname" der Bibliothek.

### Beispiel:

```
#import "kernel32.dll"
    int GetLastError();
#import "lib.ex5"
    int GetLastError();
#import

class CFoo
{
public:
    int      GetLastError() { return(12345); }
    void     func()
    {
        Print(GetLastError());           // Aufruf der Klassenmethode
        Print(::GetLastError());         // Aufruf der MQL5-Funktion
        Print(kernel32::GetLastError()); // Aufruf der Funktion kernel32.dll
        Print(lib::GetLastError());      // Aufruf der Funktion lib.ex5
    };
};

void OnStart()
{
    CFoo foo;
    foo.func();
};
```



Sehen Sie auch

[überladen](#), [Virtuelle Funktionen](#), [Polymorphismus](#)

## Exportieren der Funktionen

Es gibt die Möglichkeit im mql5-Programm die Funktion zu benutzen, erklärt im anderen mql5-Programm mit dem Postmodifikator *export*. Diese Funktion heisst Exportfunktion und sie kann aus anderen Programmen nach der Compilierung aufgerufen werden.

```
int Function() export
{
}
```

Dieser Modifikator befiehlt dem Compiler, die Funktion in die Tabelle der EX5-Funktionen einzutragen, die von der ex5 Dstei exportiert werden. Nur die Funktionen mit diesem Modifikator werden aus anderen mql5-Programmen zugaenglich ("sichtbar").

Eigenschaft [library](#) deutet dem Compiler hin, dass die vorliegende EX5-Datei Bibliothek sein wird und Compilerer stellt das in Titel EX5.

Alle Funktionen geplant als Exportfunktionen muessen von dem Modifikator *export* vermerkt werden.

Sehen Sie auch

[überladen](#), [Virtuelle Funktionen](#), [Polymorphismus](#)

## Ereignisbearbeiter

In der Sprache MQL5 ist die Bearbeitung einiger [vorbestimmten Ereignissen](#) vorgesehen. Funktionen für die Bearbeitung dieser Ereignisse müssen im Programm MQL5 bestimmt werden; Funktionsname, Typ des Rückgabewertes, Parametervorrat (wenn sie vorhanden sind) und ihre Typen müssen der Beschreibung der Funktion-Bearbeiter des Ereignisses strikt entsprechen.

Nämlich nach dem Typ des Rückgabewertes und Typen der Parameter identifiziert der Bearbeiter von Ereignissen des Kundenterminals die Funktionen, die verschiedene Ereignisse bearbeiten. Wenn die entsprechende Funktion andere angegebene Parameter hat, die den folgenden Beschreibungen nicht entsprechen oder ein anderer Typ des Rückgabewertes angegeben wird, wird diese Funktion für die Bearbeitung des Ereignisses nicht benutzt werden.

### OnStart

Funktion OnStart() ist der Bearbeiter des Ereignisses [Start](#), wird nur für ablaufende Scripts automatisch generiert. Muss den Typ **void** haben, hat keine Parameter:

```
void OnStart();
```

für die Funktion OnStart() ist es zugänglich, den Typ des Rückgabewertes int anzugeben.

### OnInit

Funktion OnInit() ist der Bearbeiter des Ereignisses [Init](#). Kann den Typ **void** oder **int**, haben, hat keine Parameter:

```
void OnInit();
```

Ereignis Init wird sofort nach Der Expert- oder Indikatorladung generiert werden, für Scripts wird dieses Ereignis nicht generiert. Funktion OnInit() wird für die Initialisierung verwendet. Wenn OnInit() Rückgabewert des Typs int hat, bedeutet der Nichtnullrückgabekode verfehlte Initialisierung und generiert Ereignis [Deinit](#) mit dem Code des Deinitialisierungsgrundes [REASON\\_INITFAILED](#).

Für Optimierung von Eingabeparametern wird es empfohlen Werte aus [ENUM\\_INIT\\_RETCODE](#) als Rückgabecode zu verwenden. Diese Werte werden für die Organisation der Verlauf der Optimierung, einschließlich der Auswahl der am besten geeigneten [Testagenten](#) eingesetzt. Während der Initialisierung eines Expert Advisor vor Beginn des Tests können Sie Informationen über die Konfiguration und die Ressourcen eines Agenten (die Anzahl der Kerne, Menge des freien Speichers, etc.) unter Verwendung des [TerminalInfoInteger\(\)](#)-Funktion anfordern. Basierend auf diesen Informationen können Sie entweder Verwendung dieses Agents erlauben oder ablehnen bei der Optimierung dieses Expert Advisors.

### ENUM\_INIT\_RETCODE

| Identifikator  | Beschreibung                                                                                                                                                                                         |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| INIT_SUCCEEDED | Erfolgreiche Initialisierung, Test des Expert Advisor kann fortgesetzt werden.<br>Dieser Code bedeutet das gleiche wie ein Null-Wert - der Expert Advisor wurde erfolgreich im Tester initialisiert. |

|                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|---------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| INIT_FAILED               | <p>Initialisierung fehlgeschlagen, es gibt keinen Sinn den Test fortzusetzen wegen der fatale Fehler. Zum Beispiel konnte nicht einen Indikator erstellen, der für die Arbeit des Expert Advisors erforderlich ist.</p> <p>Dieser Rückgabewert bedeutet das gleiche wie ein anderer Wert als Null - Initialisierung des Expert Advisors im Tester gescheitert. Test für die gegebenen Satz von Parametern des Expert Advisors nicht ausgeführt wird, ist der Agent frei, um eine neue Aufgabe zu erhalten.</p> |
| INIT_PARAMETERS_INCORRECT | <p>Mit diesem Wert kann der Programmierer einen falschen Satz von Eingabeparameter beziehen. Auf die gesamte Optimierung-Tabelle wird die Zeile des Ergebnisses diesem Return-Code in rot hervorgehoben.</p> <p>Nach Erhalt dieses Wert wird Strategie Tester garantiert diese Aufgabe an andere Agenten für erneut Ausführen nicht übertragen.</p>                                                                                                                                                            |
| INIT_AGENT_NOT_SUITABLE   | <p>Es gibt keine Fehler während der Initialisierung, aber aus irgendeinem Grund ist der Agent nicht geeignet für Test. Zum Beispiel, nicht genug Speicher, kein <a href="#">OpenCL-Unterstützung</a> etc.</p> <p>Nach der Rückkehr von diesem Code wird der Agent Aufgaben bis zum Ende <a href="#">dieser Optimierung</a> nicht empfangen.</p>                                                                                                                                                                |

Funktion OnInit() des Typs void bedeutet immer erfolgreiche Initialisierung.

## OnDeinit

Funktion OnDeinit() wird bei der Deinitialisierung aufgerufen und ist der Bearbeiter des Ereignisses [Deinit](#). Muss mit dem Typ `void` erklärt werden und einen Parameter des Typs `const int` haben, der [Kode des Deinitialisierungsgrundes](#) hat. Wenn ein anderer Typ erklärt wird, gibt... Für Scripts wird das Ereignis Deinit nicht generiert, darum darf man nicht die Funktion OnDeinit() in Scripts verwenden.

```
void OnDeinit(const int reason);
```

Ereignis Deinit wird für Experten und Indikatoren in folgenden Fällen generiert:

- vor Uminitialisierung wegen der Änderung des Symbols oder Chartperiode, zu dem mql5-Programm angeschlossen ist;
- vor Neuinitialisierung wegen der Veränderung von [Inputparametern](#);
- vor Entladung des mql5-Programms.

## OnTick

Ereignis [NewTick](#) wird nur für Experten generiert beim Eingang des neuen Ticks nach Symbol, zu

dessen Chart Expert angeschlossen ist. Es ist sinnlos, die Funktion `OnTick()` im Benutzerindikator oder Script zu bestimmen, denn das Ereignis `NewTick` wird für sie nicht generiert.

Ereignis `Tick` wird nur für Experten generiert, das bedeutet aber nicht, dass Experten die Funktion `OnTick()` haben müssen, denn für Experten werden nicht nur Ereignisse `NewTick`, sondern auch Ereignisse `Timer`, `BookEvent` und `ChartEvent` generiert. Muss mit dem Typ `void`, erklärt werden hat keine Parameter:

```
void OnTick();
```

## OnTimer

Funktion `OnTimer()` wird beim Erscheinen des Ereignisses [Timer](#) aufgerufen, das vom Systemtimer nur für Experten und Anzeiger generiert wird - in Skripts kann sie nicht verwendet werden. Funktion `OnTimer()` wird nur beim Eintritt des Ereignisses [Timer](#) aufgerufen, das vom Systemtimer nur für Experten generiert wird - man muss sie nicht in Scripts oder Indikatoren verwenden. Periode des Eintritt dieses Ereignisses wird bei der Subskription auf Benachrichtigungsempfang über dieses Ereignis durch die Funktion [EventSetTimer\(\)](#) eingesetzt.

Annulieren der Aufnahme der Timerereignisse für jeden konkreten Experten erfolgt durch die Funktion [EventKillTimer\(\)](#). Funktion muss mit dem Typ `void` bestimmt werden, hat keine Parameter:

```
void OnTimer();
```

Es ist empfehlenswert, die Funktion `EventSetTimer()` in der Funktion `OnInit()` einmalig aufzurufen, und Die Funktion `EventKillTimer()` einmalig in der Funktion `OnDeinit()` aufzurufen.

Jeder Expert und Anzeiger arbeitet nur mit seinem Timer und erhält Ereignisse nur von ihm. Wenn das mql5-Programm zu funktionieren stoppt, wird Timer zwangsläufig entfernt, wenn er durch die Funktion [EventKillTimer\(\)](#) erzeugt aber nicht entfernt wurde.

## OnTrade

Funktion wird beim Eintritt von [Trade](#) aufgerufen, das bei der Änderung der Liste der [aufgestellten Order](#) und [offenen Positionen](#), [Ordergeschichte](#) und [Dealgeschichte](#) entsteht. Bei jeder Handelshandlung (Aufstellung der zurückgesetzten Order, Öffnung/Schließung der Position, Einstellung von Stops, Auslösung der zurückgesetzten Ordern usw.) verändert sich entsprechend die Geschichte der Ordern und Deals und/oder Liste der Positionen und laufender Ordern.

```
void OnTrade();
```

Benutzer muss selbstständig, Prüfung des Standes vom Handelskonto beim Erhalten dieses Ereignisses zu realisieren (wenn es nach Bedingungen der Handelsstrategie notwendig ist). Wenn der Funktionsaufruf `OrderSend()` erfolgreich beendet und den Wert `true` zurückgibt - bedeutet es, dass Handelsserver Order in Warteschlange eingereicht hat und ihr die Ticketnummer zugeordnet hat. Sobald der Server diesen Befehl bearbeitet, wird das Ereignis `Trade` generiert werden. Wenn der Benutzer den Ticketwert im Kopf behält, kann er bei der Bearbeitung des Ereignisses `OnTrade()` aufklären, was mit der Order geschehen ist.

## OnTradeTransaction

Als Ergebnis der Durchführung von bestimmten Aktionen auf dem Handelskonto, ändert sich der Kontostand. Zu solchen Aktionen gehören:

- Senden der Handelsanfrage aus einer beliebigen MQL5-Anwendung in dem Kundenterminal mittels der Funktionen [OrderSend](#) und [OrderSendAsync](#) und ihre nachfolgende Ausführung;
- Senden der Handelsanfrage über die grafische Oberfläche des Terminals und ihre nachfolgende Ausführung;
- Auslösen von aufgeschobenen Ordnern und Stop-Ordnern auf dem Server;
- Ausführen von Operationen auf der Seite der Handel-Server.

Als Ergebnis dieser Aktionen werden die folgenden Handelstransaktionen durchgeführt:

- Die Verarbeitung der Handelsanfrage;
- Änderung der offenen Aufträge;
- Änderung der Ordergeschichte;
- Änderung der Dealgeschichte;
- Änderung der Position.

Zum Beispiel, beim Senden Marktkauforder, wird Marktkauforder behandelt und eine entsprechende Marktkauforder für das Konto erstellt. Ist die Order durchgeführt, wird sie aus der Auftragsliste entfernt und zur Ordergeschichte hinzugefügt. Dann wird der entsprechende Deal zur Geschichte hinzugefügt und eine neue Position wird erstellt. Alle diese Aktionen sind Handelstransaktionen. Ankunft jeder solchen Transaktion im Terminal ist ein Ereignis [TradeTransaction](#). Es ruft den Handler `OnTradeTransaction` auf

```
void OnTradeTransaction(
    const MqlTradeTransaction& trans,      // Struktur der Handelstransaktion
    const MqlTradeRequest& request,        // Struktur der Anfrage
    const MqlTradeResult& result           // Struktur der Antwort
);
```

Der Handler enthält drei Parameter:

- **trans** - in diesen Parameter wird die Struktur [MqlTradeTransaction](#) übergeben, die die auf das Handelskonto angewendete Handelstransaktion beschreibt;
- **request** - in diesen Parameter wird die Struktur [MqlTradeRequest](#) übergeben, die eine Handelsanfrage beschreibt;
- **result** - in diesen Parameter wird die Struktur [MqlTradeResult](#) übergeben, die das Ergebnis der Ausführung einer Handelsanfrage beschreibt.

Zwei letzte Parameter **request** und **result** werden mit den Werten nur für Transaktion des Typs [TRADE\\_TRANSACTION\\_REQUEST](#) ausgefüllt, die Information über Transaktion kann man aus dem Parameter *type* der Variable **trans** erhalten. Beachten Sie, dass in diesem Fall das Feld *request\_id* in der Variable **result** den Identifikator [der Handelsanfrage request](#) enthält, als Ergebnis dessen Ausführung die in der Variable **trans** beschriebene **Handelstransaktion** durchgeführt wurde. Das Vorhandensein des Identifikators der Anfrage erlaubt die ausgeführte Aktion (Aufruf der Funktion `OrderSend` oder `OrderSendAsync`) mit dem in die [OnTradeTransaction\(\)](#) übergebenen Ergebnis dieser Aktion zu verbinden.

Eine Handelsanfrage, die aus dem Terminal manuell oder durch die Handelsfunktionen [OrderSend\(\)](#)/[OrderSendAsync\(\)](#) gesendet ist, kann mehrere konsequente Handelstransaktionen auf dem Handel-Server auslösen. Dabei ist die Reihenfolge des Eingangs dieser Transaktionen in das Terminal nicht gewährleistet, deshalb darf man seinen Handelsalgorithmus auf der Erwartung des Eingangs einiger Handelstransaktionen nach der Ankunft der anderen nicht bauen. Außerdem können Transaktionen während der Übergabe vom Server zum Terminal verloren werden.

- Alle Typen der Handelstransaktionen werden in der Aufzählung [ENUM\\_TRADE\\_TRANSACTION\\_TYPE](#) beschrieben.
- Die Struktur `MqlTradeTransaction`, die eine Handelstransaktion beschreibt, wird auf unterschiedlicher Weise je nach Typ der Transaktion ausgefüllt. Zum Beispiel, für Transaktionen des Typs `TRADE_TRANSACTION_REQUEST` muss nur `type` Feld (Typ der Handelstransaktion) analysiert werden. Für zusätzliche Informationen muss man die zweiten und dritten Parameter der Funktion `OnTradeTransaction` (`request` und `result`) analysieren. Weitere Informationen finden Sie im Abschnitt "[Struktur der Handelstransaktion](#)".
- In der Beschreibung der Handelstransaktion werden nicht alle verfügbaren Informationen über Aufträge, Deals und Positionen (zum Beispiel, Kommentar) übergeben. Für erweiterte Informationen sollten die Funktionen [OrderGet\\_\\*](#), [HistoryOrderGet\\_\\*](#), [HistoryDealGet\\_\\*](#) und [PositionGet\\_\\*](#) verwendet werden.

Nach der Anwendung der Handelstransaktionen auf das Kundenkonto werden sie in die Warteschlange der Handelstransaktionen des Terminals konsequent gestellt, woher in den Eingangspunkt `OnTradeTransaction` in der Reihenfolge der Ankunft in das Terminal schon konsequent übergeben werden.

Während der Verarbeitung der Handelstransaktionen von einem Expert Advisor mit Hilfe des Handlers `OnTradeTransaction` bearbeitet das Terminal neu ankommenden Handelstransaktionen weiter. Somit kann sich der Zustand des Handelskontodes während des `OnTradeTransaction` schon ändern. Zum Beispiel, während das MQL5-Programm ein Ereignis des Hinzufügens einer neuen Order bearbeitet, kann sie ausgeführt werden, aus der offenen Auftragsliste entfernt werden und in die Geschichte übertragen werden. Im Folgenden wird das Programm über alle diesen Ereignisse benachrichtigt sein.

Länge der Warteschlange umfasst 1024 Elemente. Falls `OnTradeTransaction` die nächste Transaktion zu lange bearbeiten wird, können die alten Transaktionen in der Warteschlange durch die neueren verdrängt sein.

- Im Allgemeinen gibt es kein genaues Verhältnis nach der Anzahl der Aufrufe `OnTrade` und `OnTradeTransaction`. Ein Anruf `OnTrade` entspricht einem oder mehreren Aufrufen `OnTradeTransaction`.
- `OnTrade` wird nach den entsprechenden Aufrufen `OnTradeTransaction` aufgerufen.

## OnTester

Funktion `OnTester()` ist Bearbeiter des Ereignisses [Tester](#), das nach Abschluss des historischen Testen des Experten für das vorgegebene Datenintervall. Funktion muss mit dem Typ `void` bestimmt werden, hat keine Parameter:

```
double OnTester();
```

Funktion wird unmittelbar vor dem Aufruf der Funktion `OnDeinit()` aufgerufen und hat den Typ des Rückgabewertes `double`. Funktion `OnTester()` kann nur in Experten beim Testen verwendet werden und ist in erster Linie für Berechnung bestimmtes Wertes bestimmt, der als Kriterium `Custom max` bei genetischer Optimierung der Eingabeparameter verwendet.

Bei genetischer Optimierung wird Sortieren der Ergebnisse innerhalb einer Generation absteigend durchgeführt. D.h. die besten Ergebnisse vom Standpunkt des Optimierungskriterium sind die mit dem größten Wert (für Optimierungskriterium `Custom max` werden Werte beachtet, die durch die Funktion

OnTester zurückgegeben werden). Die schlechtesten Werte bei diesem Sortieren werden am Ende gestellt und danach weggeworfen und an Formieren der nächsten Generation nicht beteiligen.

## OnTesterInit

Die Funktion OnTesterInit() ist ein Eventhandler von [TesterInit](#), das automatisch vor dem Start der Optimierung eines Expert Advisors generiert wird. Die Funktion muss mit dem Typ void definiert werden, ohne Parameter:

```
void OnTesterInit();
```

Ein Expert Advisor, der den Handler OnTesterDeinit() oder OnTesterPass() hat, wird beim Start der Optimierung automatisch auf einem gesonderten Chart des Terminals geladen mit dem Symbol und der Periode, die im Tester angegeben wurden, und ein Ereignis TesterInit empfängt. Die Funktion soll einen Expert Advisor vor Optimierung initialisieren um dann [die Ergebnisse der Optimierung zu verarbeiten](#).

## OnTesterPass

Die Funktion OnTesterPass() ist ein Eventhandler von [TesterPass](#), das automatisch vor dem Start der Optimierung eines Expert Advisors generiert wird. Die Funktion muss mit dem Typ void definiert werden, ohne Parameter:

```
void OnTesterPass();
```

Ein Expert Advisor mit Handler OnTesterPass() wird automatisch auf einem gesonderten Chart des Terminals geladen mit dem für ein Test angegebenen Symbol und der Periode, und erhält Ereignisse von TesterPass beim Erhalten von einem Frame bei der Optimierung. Die Funktion wird für die dynamische Verarbeitung von [Optimierungsergebnisse](#) direkt "on the fly" ausgelegt, ohne für die Vollendung der Optimierung zu warten. Frames werden durch Funktion [FrameAdd\(\)](#) hinzugefügt. Diese Funktion kann nach einem Durchgang im Handler [OnTester\(\)](#) [aufgerufen werden](#).

## OnTesterDeinit

Die Funktion OnTesterDeinit() ist ein Eventhandler von [TesterDeinit](#), das automatisch nach der Optimierung eines Expert Advisors generiert wird. Die Funktion muss mit dem Typ void definiert werden, ohne Parameter:

```
void OnTesterDeinit();
```

Ein Expert Advisor mit dem Handler OnTesterDeinit() wird automatisch auf dem Chart beim Start der Optimierung geladen, und erhält ein Ereignis TesterDeinit nach ihrer Fertigstellung. Die Funktion wird für die abschließende Bearbeitung aller [Optimierungsergebnisse](#) verwendet.

## OnBookEvent

Funktion OnBookEvent() ist der Bearbeiter von Ereignis [BookEvent](#). Ereignis BookEvent wird nur für Experten und Anzeiger bei der Veränderung von DOM (Depth of Market generiert). Muss den Typ void und einen Parameter des Typs string haben:

```
void OnBookEvent (const string& symbol);
```

Um Ereignisse BookEvent für jedes Symbol zu bekommen, reicht es, voraussichtlich auf Erhalten dieser Ereignisse für dieses Symbol durch die Funktion [MarketBookAdd\(\)](#) zu subscribieren. Um die



Subskription auf Erhalten des Ereignisses BookEvent für konkretes Symbol zu annullieren, muss man die Funktion [MarketBookRelease\(\)](#) aufrufen.

Zum Unterschied von den anderen Ereignissen, ist das Ereignis BookEvent broadcast. Das bedeutet, wenn ein Expert Auf Erhalten des Ereignisses BookEvent durch die Funktion MarketBookAdd subskribiert, werden alle anderen Experten mit Bearbeiter OnBookEvent() dieses Ereignis bekommen. Darum muss der Symbolname, der in Bearbeiter als Parameter *const string& symbol* übertragen wird, analysieren.

## OnChartEvent

OnChartEvent() ist der Bearbeiter von Gruppe der Ereignisse [ChartEvent](#):

- CHARTEVENT\_KEYDOWN – Ereignis des Keyboardbetätigens, wenn das Chartfenster im Fokus steht;
- CHARTEVENT\_MOUSE\_MOVE – Ereignisse der Maus-Bewegung und Mausklicks (wenn die Eigenschaft [CHART\\_EVENT\\_MOUSE\\_MOVE](#)=true ist eingegeben);
- CHARTEVENT\_OBJECT\_CREATE – Ereignis der Erzeugung des graphischen Objektes (wenn die Eigenschaft [CHART\\_EVENT\\_OBJECT\\_CREATE](#)=true ist eingegeben);
- CHARTEVENT\_OBJECT\_CHANGE – Veränderung der Objekteigenschaften durch Dialog der Eigenschaften;
- CHARTEVENT\_OBJECT\_DELETE – Ereignis der Entfernung des graphischen Objektes (wenn die Eigenschaft [CHART\\_EVENT\\_OBJECT\\_DELETE](#)=true ist eingegeben);
- CHARTEVENT\_OBJECT\_CLICK – Ereignis des Mausklickens auf dem graphischen Objekt, das zum Chart gehört;
- CHARTEVENT\_OBJECT\_DRAG – Ereignis der Übertragung des graphischen Objekts mit der Maus;
- CHARTEVENT\_OBJECT\_ENDEDIT – Ereignis des Abschlusses von Texteditieren im Eingabefeld des graphischen Objekts LabelEdit;
- CHARTEVENT\_CHART\_CHANGE – Ereignis von Änderungen des Charts;
- CHARTEVENT\_CUSTOM+n – Identifikator des Benutzerereignisses, wo sich n im Bereich von 0 bis 65535 befindet.
- CHARTEVENT\_CUSTOM\_LAST – letzter zugeänglicher Identifikator des Benutzerereignisses (CHARTEVENT\_CUSTOM+65535).

Funktion kann nur in Experten und Indikatoren aufgerufen werden und muss den Typ void und 4 Parameter haben:

```
void OnChartEvent(const int id,          // Identifikator des Ereignisses
                  const long& lparam,    // Parameter des Ereignisses des Typs long
                  const double& dparam,  // Parameter des Ereignisses des Typs double
                  const string& sparam   // Parameter des Ereignisses des Typs string
                  );
```

Für jeden Typ des Ereignisses haben Eingabeparameter der Funktion OnChartEvent() vorbestimmte Werte, die für Verarbeitung dieses Ereignisses notwendig sind. In der Tabelle sind Ereignisse und Werte aufgezählt, die durch Parameter übertragen werden.

| Ereignis | Wert des Parameters id | Wert des Parameters | Wert des Parameters | Wert des Parameters |
|----------|------------------------|---------------------|---------------------|---------------------|
|          |                        |                     |                     |                     |

|                                                                                                                                                      |                          | lparam                    | dparam                                                                                              | sparam                                                                         |
|------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------|---------------------------|-----------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------|
| Ereignis des Keyboard Klickens                                                                                                                       | CHARTEVENT_KEYDOWN       | Kode der betätigten Taste | Die Anzahl der Tastenanschläge, die generiert werden, während diese Taste in gedrücktem Zustand war | Der String-Wert eines Bit-Maske, die den Status der Tastatur-Tasten beschreibt |
| Ereignisse der Maus-Bewegung und Mausklicks (wenn die Eigenschaft <a href="#">CHART_EVENT_MOUSE_MOVE</a> =true ist für das Chart eingegeben)         | CHARTEVENT_MOUSE_MOVE    | X-Koordinate              | Y-Koordinate                                                                                        | Der String-Wert einer Bit-Maske, der den Status der Maustasten beschreibt      |
| Ereignis der Erzeugung des graphischen Objektes (wenn die Eigenschaft <a href="#">CHART_EVENT_OBJECT_CREATE</a> =true ist für ein Chart eingegeben)  | CHARTEVENT_OBJECT_CREATE | —                         | —                                                                                                   | Name des erzeugten graphischen Objektes                                        |
| Ereignis der Veränderung der Eigenschaften des Objektes durch Dialog der Eigenschaften                                                               | CHARTEVENT_OBJECT_CHANGE | —                         | —                                                                                                   | Name des veränderten graphischen Objektes                                      |
| Ereignis der Entfernung des graphischen Objektes (wenn die Eigenschaft <a href="#">CHART_EVENT_OBJECT_DELETE</a> =true ist für ein Chart eingegeben) | CHARTEVENT_OBJECT_DELETE | —                         | —                                                                                                   | Name des entfernten graphischen Objektes                                       |
| Ereignis des Mausklicks auf der graphischen Darstellung                                                                                              | CHARTEVENT_CLICK         | X-Koordinate              | Y-Koordinate                                                                                        | —                                                                              |

|                                                                                                  |                           |                                                                        |                                                                        |                                                                                        |
|--------------------------------------------------------------------------------------------------|---------------------------|------------------------------------------------------------------------|------------------------------------------------------------------------|----------------------------------------------------------------------------------------|
| Ereignis des Mausklicks auf dem graphischen Objekt                                               | CHARTEVENT_OBJECT_CLICK   | X-Koordinate                                                           | Y-Koordinate                                                           | Name des graphischen Objektes, auf dem das Ereignis geschehen ist                      |
| Ereignis der Verschiebung des graphischen Objektes per Mausklick                                 | CHARTEVENT_OBJECT_DRAG    | —                                                                      | —                                                                      | Name des verschobenen graphischen Objektes                                             |
| Ereignis der Beendigung des Texteditierens im Eingabefeld des graphischen Objektes "Eingabefeld" | CHARTEVENT_OBJECT_ENDEDIT | —                                                                      | —                                                                      | Name des graphischen Objektes "Eingabefeld", in dem Editieren des Textes beendet wurde |
| Ereignis der Chart-Veränderung                                                                   | CHARTEVENT_CHART_CHANGE   | —                                                                      | —                                                                      | —                                                                                      |
| Benutzerereignis mit Nummer N                                                                    | CHARTEVENT_CUSTOM+N       | Wert, vorgegeben durch die Funktion <a href="#">EventChartCustom()</a> | Wert, vorgegeben durch die Funktion <a href="#">EventChartCustom()</a> | Wert, vorgegeben durch die Funktion <a href="#">EventChartCustom()</a>                 |

## OnCalculate

Funktion OnCalculate() wird nur in Benutzerindikatoren aufgerufen werden bei der Notwendigkeit, Berechnung der Indikatorenwerte am Ereignis [Calculate](#) durchzuführen. Gewöhnlich erfolgt es bei der Aufnahme des neuen Ticks für Symbol, für das der Indikator berechnet wird. Dabei braucht der Indikator nicht zum einen Preischart dieses Symbols angehängt werden.

Funktion OnCalculate() muss den Typ des Rückgabewertes int haben. Es gibt zwei Definitionsvarianten. Innerhalb eines Indikators können beide Varianten der Funktion nicht verwendet werden.

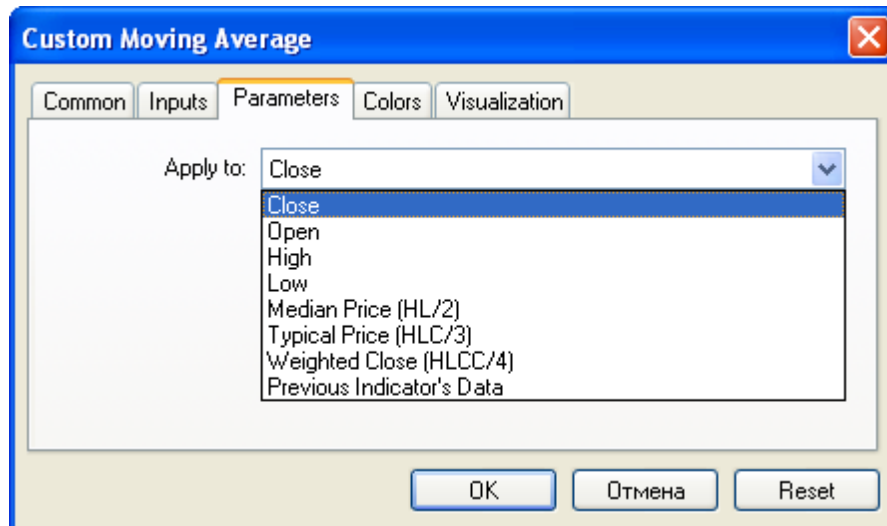
Die erste Aufrufform ist für die Indikatoren bestimmt, die auf einem Datenpuffer berechnet werden können. Das Beispiel dieses Indikators - Custom Moving Average.

```
int OnCalculate (const int rates_total,      // Feldgroesse price[]
                const int prev_calculated,  // verarbeitete Bars im früheren Aufruf
                const int begin,           // woher bedeutsame Daten anfangen
                const double& price[]      // Feld für Berechnung
                );
```

Als Feld price[] kann eine der Preistimeserien oder berechneter Buffer irgendwelches Anzeigers übertragen werden. Um Indizierungsrichtung im Feld price[] zu bestimmen, muss man die Funktion

[ArrayGetAsSeries\(\)](#) aufrufen. Um von Default-Werten nicht abzuhängen, muss man die Funktion [ArraySetAsSeries\(\)](#) für die Felder, mit denen es zu arbeiten zu erwarten, aufrufen.

Auswahl der notwendigen Timeserie oder des Indikators als Feld price[] macht der Benutzer beim Start des Indikators auf Registerblatt "Parameters". Dafür muss das notwendige Element in der drop-down Liste des Feldes "Apply to" angegeben werden.



für Erhalten der Werte des Benutzerindikators aus anderen mql5-Programmen wird die Funktion [iCustom\(\)](#) verwendet, , die handle des Indikators für folgende Operationen rückgibt. Dabei kann auch das notwendige Feld price[] oder handle des anderen Indikators angegeben werden. Dieser Parameter muss als letzte in der Liste von Variablen des Benutzerindikators übertragen werden.

**Beispiel:**

```
void OnStart()
{
//---
string terminal_path=TerminalInfoString(STATUS_TERMINAL_PATH);
int handle_customMA=iCustom(Symbol(),PERIOD_CURRENT, "Custom Moving Average",13,0,
if(handle_customMA>0)
Print("handle_customMA = ",handle_customMA);
else
Print("Cannot open or not EX5 file '"+terminal_path+"\\MQL5\\Indicators\\"+"Cust
; }
```

In diesem Beispiel ist der letzte Parameter der Wert PRICE\_TYPICAL (aus Aufzählung [ENUM\\_APPLIED\\_PRICE](#)), der deutet, dass Benutzerindikator auf typischen Preisen gebaut werden wird, erhalten als (High+Low+Close)/3. Wenn der Parameter nicht angegeben wird, wird Indikator auf Grundlage der Werte PRICE\_CLOSE gebaut,d.h. auf Schlusspreisen jede Bar.

Ein anderes Beispiel das übergabe des handles des Indikators als letzter Parameter demonstriert für Festlegung des Feldes price[], ist in der Funktionsbeschreibung [iCustom\(\)](#) angegeben.

Die zweite Aufrufform ist für alle anderen Indikatoren bestimmt, die für Berechnung mehr als eine Zeitreihe verwenden.

```
int OnCalculate (const int rates_total, // Größe der Eingabezeitreihen
const int prev_calculated, // Bars, verarbeitete im früheren Aufruf
```

```

const datetime& time[],      // Time
const double& open[],        // Open
const double& high[],        // High
const double& low[],         // Low
const double& close[],       // Close
const long& tick_volume[],   // Tick Volume
const long& volume[],        // Real Volume
const int& spread[]          // Spread
);

```

Parameter open[], high[], low[] und close[] enthalten Felder mit Öffnungspreisen, dem maximalen und minimalen Preis und Schlusspreisen des laufenden Timeframes. Parameter time [] enthält Feld mit Eröffnungszeit, Parameter spread[] - Feld mit der Geschichte der Spreads (wenn Spread für diesen Typ (wenn Spread für dieses Handelsinstrument vorausgesehen wird). Parameter volume[] und tick\_volume[] enthalten die Geschichte des Handels- und Tickvolumen.

Zu bestimmen die Richtung des Indizierens in Feldern time[], open[], high[], low[], close[], tick\_volume[], volume[] und spread[], muss man die Funktion [ArrayGetAsSeries\(\)](#) aufrufen. Damit man von Default-Werten nicht abhängt, muss man die Funktion [ArraySetAsSeries\(\)](#) für die Felder aufrufen, mit denen es zu arbeiten erwartet.

Der erste Parameter rates\_total enthält die Zahl von Bars, zugängliche dem Indikator für Berechnungsrund entspricht der Baranzahl, zugängliche auf dem Chart.

Wir müssen über den Zusammenhang zwischen dem Rückgabewert von OnCalculate() und dem zweiten Eingabeparameter prev\_calculated erwähnen. Parameter prev\_calculated enthält beim Funktionsaufruf den Wert, den die Funktion OnCalculate() beim **früheren** Aufruf **rückgegeben** hat. Das ermöglicht rationale Berechnungsalgorithmen des Benutzerindikators zu realisieren, um Wiederberechnungen für die Bars, die seit früheren Start dieser Funktion nicht verändert haben, zu vermeiden.

Dafür reicht es gewöhnlich, Parameterwert rates\_total rückzugeben, der die Baranzahl beim laufenden Funktionsaufruf enthält. Wenn vom Moment des letzten Aufrufes der Funktion OnCalculate() Preisdaten nicht verändert wurden (tiefere Geschichte heruntergeladen sind oder Auslassungen der Geschichte ausgefüllt sind), wird der Wert des Eingabeparameters prev\_calculated vom Terminal Null gesetzt werden.

**Bemerkung:** Wenn die Funktion OnCalculate Nullwert rückgibt, werden die Werte des Indikators im Fenster DataWindow des Kundenterminals nicht gezeigt.

für bessere Verständigung, ist es empfehlenswert, Indikator ablaufen zu lassen, dessen Code unten angegeben wird.

#### Beispiel des Indikators:

```

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots 1
//---- plot Line
#property indicator_label1 "Line"
#property indicator_type1 DRAW_LINE
#property indicator_color1 clrDarkBlue
#property indicator_style1 STYLE_SOLID

```

```

#property indicator_width1 1
//--- indicator buffers
double          LineBuffer[];
//+-----+
//| Custom indicator initialization function          |
//+-----+
int OnInit()
{
//--- indicator buffers mapping
    SetIndexBuffer(0,LineBuffer,INDICATOR_DATA);
//---
    return(INIT_SUCCEEDED);
;}
//+-----+
//| Custom indicator iteration function              |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime& time[],
                const double& open[],
                const double& high[],
                const double& low[],
                const double& close[],
                const long& tick_volume[],
                const long& volume[],
                const int& spread[])
{
//--- bekommen die Anzahl der zugeänglichen Bars für laufendes Symbol und laufende Per
    int bars=Bars(Symbol(),0);
    Print("Bars = ",bars," rates_total = ",rates_total," prev_calculated = ",prev_calculated);
    Print("time[0] = ",time[0]," time[rates_total-1] = ",time[rates_total-1]);
//--- return value of prev_calculated for next call
    return(rates_total);
;}
//+-----+

```

Sehen Sie auch

[Programmausführung](#), [Ereignisse des Kundenterminals](#), [Arbeit mit Ereignissen](#)

## Variablen

### Erklärung der Variablen

Variablen müssen vor der Verwendung erklärt werden. Für die Identifizierung der Variablen werden unikale Namen verwendet. Beschreibungen der Variablen werden für ihre Definition und Typenerklärung verwendet. Beschreibung ist keine Anweisung.

Einfache Typen sind:

- char, short, int, long, uchar, ushort, uint, ulong - Ganzzahlen;
- color - Ganzzahl, repräsentiert RGB-Farbe;
- datetime - Datum und Zeit, zeichenlose Ganzzahl, enthält Sekundenzahl, die seit 0 Uhr 1 Januar 1970 vergangen sind;
- bool - logische Werte true und false;
- double - Zahlen doppelter Genauigkeit mit Fließpunkt;
- float - Zahlen mit einfacher Genauigkeit mit Fließpunkt;
- string - Symbolzeilen.

Beispiele:

```
string szInfoBox;  
int     nOrders;  
double dSymbolPrice;  
bool    bLog;  
datetime tBegin_Data   = D'2004.01.01 00:00';  
color    cModify_Color = C'0x44,0xB9,0xE6';
```

### Zusammengesetzte Typen:

Strukturen - sind zusammengesetzte Datentypen, gebildet mit Verwendung von anderen Typen.

```
struct MyTime  
{  
    int hour;    // 0-23  
    int minute;  // 0-59  
    int second;  // 0-59  
};  
...  
MyTime strTime; // Variable des Typs der früher erklärten Struktur MyTime
```

Vor der Strukturerklärung kann man nicht Variablen des Typs Strukturen erklären.

### Felder

Feld - indizierte Menge gleichartiger Daten:

```
int    a[50];        // eindimensionales Feld aus 50 Ganzzahlen.  
double m[7][50];     // zweidimensionales Feld aus sieben Feldern,  
                    // jedes von denen aus 50 Zahlen besteht.
```

```
MyTime t[100];           // Feld mit den Elementen der Art MyTime
```

Feldindex kann nur Ganzzahl sein. Es werden nicht mehr als vierdimensionale Felder zugelassen. Nummerierung der Feldelementen faengt mit Null an. Das letzte Element des eindimensionalen Feldes ist um eine Nummer kleiner, als die Feldgroesse, d.h. der Zugriff zum letzten Element des Feldes aus 50 Ganzzahlen wird als `a[49]` aussehen. Dasselbe bezieht sich auf vieldimensionale Felder - Indizieren einer Dimension erfolgt von 0 bis -1. Das letzte Element des zweidimensionalen Feldes aus dem Beispiel wird als `m[6][49]` aussehen.

Statistische Felder können nicht als Timeserien dargestellt werden, d.h. die Funktion [ArraySetAsSeries\(\)](#), die den Zugang zu Feldelementen vom Ende zum Anfang des Feldes festlegt, ist für sie nicht anwendbar. Wenn es notwendig ist, den Zugang zum Feld in [Timeserien](#) zu gewaehrleisten, benutzen Sie [Objekt des dynamischen Feldes](#).

Beim Zugang ausser Feld generiert das vollziehende System kritisches Fehler und Durchführung des Programms wird unterbrochen.

## Zugangsspezifikatoren

Zugangsspezifikatoren deuten dem Compiler darauf, wie der Zugang zu Variable, Struktur- und Klassen-gliedern durchgeführt werden kann.

Spezifikator **const** erklärt die Variable als Konstante und erlaubt nicht den Wert dieser Variable bei der Programmdurchführung zu verändern. Es wird einmalige Initialisierung der Variable bei ihrer Erklarung zugelassen.

### Beispiel

```
int OnCalculate (const int rates_total,      // Feldgroesse price[]
                 const int prev_calculated, // Anzahl der verarbeiteter Bars im vorigen
                 const int begin,           // Anfangsposition der bedeutsamen Daten
                 const double& price[]      // Feld für Berechnung
                );
```

für Zugang zu Struktur- und Klassenglieder werden folgende Spezifikatoren verwendet:

- **public** - erlaubt unbegrenzten Zugang zur Variable oder Klassenmethoden;
- **protected** - erlaubt Zugang seitens Methoden der vorliegenden Klasse und seitens Methoden der [oeffentlich heritierten](#) Klassen. Anderer Zugang ist unmoeglich;
- **private** - ermöglicht Zugang zu Variablen und Klassenmethoden ausschliesslich aus Methoden der vorliegenden Klasse.
- **virtual** - nur für Klassenmethoden anwendbar (nicht aber für Strukturmethode) und meldet dem Compiler an, dass diese Methode in der Tabelle virtueller Klassenfunktionen aufgestellt werden muss.

## Speicherklassen

Es gibt drei Speicherklassen: **static**, **input** und **extern**. Diese Modifikatoren der Speicherklasse deuten dem Compiler darauf, dass entsprechende Variablen im vorbestimmten Speicherplatz eingeordnet werden, der als globaler Pool bezeichnet wird. Dabei deuten diese Modifikatoren auf besondere Bearbeitung dieser Variablen hin.



Wenn eine Variable, die auf dem lokalen Niveau erklärt wurde, nicht [statisch](#) ist, erfolgt Speicherverteilung für diese Variable automatisch auf dem Programmstack. Speicher, der nicht für ein statisches Feld verteilt wurde, erfolgt auch automatisch beim Ausgang vom Sichtbarkeitsbereich des Blocks, in dem das Feld erklärt wurde.

#### Sehen Sie auch

[Datentypen](#), [Inkapsulation und Erweiterungsfaehigkeit der Typen](#), [Initialisierung der Variablen](#), [Sichtbereich und Lebensdauer der Variablen](#), [Erzeugung und Entfernung der Objekte](#), [Statische Klassenmitglieder](#)

## Lokale Variablen

Variable, die innerhalb einer [Funktion](#) erklärt wird ist lokal. Sichtbereich der lokalen Variable ist von Funktion begrenzt, innerhalb deren sie erklärt wird. Lokale Variable kann [initialisiert](#) werden mittels jedes [Ausdrucks](#). Initialisierung der lokalen Variable wird jedesmal beim Aufruf entsprechender Funktion durchgeführt. Lokale Variablen werden im zeitlichen Speicherbereich entsprechender Funktion aufgestellt.

### Beispiel:

```
int somefunc()
{
    int ret_code=0;
    ...
    return(ret_code);
; }
```

Operationsbereich (oder [Sichtbereich](#)) der Variable ist Programmteil, in dem auf die Variable verwiesen werden kann. Variable, die innerhalb des Blocks erklärt werden (auf innerem Niveau) haben [Block](#) als Operationsbereich. Operationsbereich Block faengt mit der Erklarung der Variable und beendet mit der rechten geschweiften Klammer.

Lokale Variablen, erklärt am Funktionsanfang haben Operationsbereich Block sowie [Funktionsparameter](#), die lokale Variablen sind. Jeder Block kann Variablenerklärungen enthalten. Wenn Blocks eingebettet sind und [Identifikator](#) im Aussenblock den selben Namen, wie Identifikator im innenblock hat, ist der Identifikator des Aussenblocks "unsichtbar" (verhüllt) bis Fertigstellung des Innenblocks.

### Beispiel:

```
void OnStart()
{
    //---
    int i=5;      // lokale Variable der Funktion
    {
        int i=10; // Variable der Funktion
        Print("im Block i = ",i); // Ergebnis i=10;
    ; }
    Print("ausser Block i = ",i); // Ergebnis i=5;
; }
```

Das bedeutet, dass sobald der Innenblock durchgeführt wird, sieht er die Werte seiner eigenen lokalen Identifikatoren, und nicht die Werte der Identifikatoren mit identischen Namen im umfassenden Block.

### Beispiel:

```
void OnStart()
{
    //---
    int i=5;      // lokale Variable der Funktion
    for(int i=0;i<3;i++)
```

```
Print("Innerhalb for i = ",i);  
Print("Ausserhalb i = ",i);  
;}  
/* Ergebnis der Durchführung  
innerhalb for i = 0  
innerhalb for i = 1  
innerhalb for i = 2  
ausserhalb des Blocks i = 5  
*/
```

Lokale Variablen erklärt als [static](#), haben Operationsbereich Block, obwohl sie seit Anfang der Programmdurchführung existieren.

## Stack

In jedem MQL5 Programm, ist ein spezieller Speicherbereich genannt zur Speicherung von automatisch erstellten lokalen variable Funktionen verteilt. Ein Stack wird für alle Funktionen reserviert, die Standard-Stack-Größe beträgt 256 KB. Stack-Größe kann durch Compiler-Direktive [#property stacksize](#) verwaltet werden.

[Statische](#) lokale Variablen werden an der gleichen Stelle, wo andere statische und [globale](#) Variablen gespeichert werden gespeichert - In einem speziellen Speicherbereich, der separat aus dem Stack besteht. [Dynamisch](#) erstellte Variablen auch verwenden ein separater Speicherbereich (getrennt von der Stack).

Mit jedem Aufruf der Funktion, wird ein Platz auf dem Stack für interne nicht-statische Variablen zugewiesen. Nach dem Verlassen der Funktion ist der Speicher wieder verfügbar.

Wenn aus der ersten Funktion die zweite Funktion aufgerufen wird, dann nimmt die zweite Funktion die erforderliche Größe aus dem verbleibenden Stack-Speicher für ihre Variablen. Damit bei der Verwendung von enthaltenden Funktionen, wird Stack-Speicher sequentiell für jede Funktion belegt werden. Dies kann zu einem Mangel an Arbeitsspeicher während einer der Funktionsaufrufe führen, ist eine solche Situation Stack-Überlauf genannt.

Daher ist für große lokale Daten besser den dynamischen Speicher zu verwenden - bei der Eingabe einer Funktion, den Speicher für die lokalen Bedürfnisse im System zu reservieren ([new](#), [ArrayResize\(\)](#)), und beim Verlassen der Funktion den Speicher freizugeben ([delete](#), [ArrayFree\(\)](#)).

Sehen Sie auch

[Datentypen](#), [Inkapsulation und Erweiterungsfaehigkeit der Typen](#), [Initialisierung der Variablen](#), [Sichtbereich und Lebensdauer der Variablen](#), [Erzeugung und Entfernung der Objekte](#)

## Formale Parameter

Die in Funktion übertragenden Parameter sind lokal. Sichtbereich ist Funktionsblock. Formale Parameter müssen sich durch Namen von Aussenparametern und lokalen Variablen unterscheiden, bestimmt innerhalb der Funktion. Im Funktionsblock können formalen Parametern bestimmte Werte zugeordnet werden. Wenn der formale Parameter mit dem Modifikator const erklärt wird, kann sein Wert innerhalb der Funktion nicht verändert werden.

### Beispiel:

```
void func(const int & x[], double y, bool z)
{
    if(y>0.0 && !z)
        Print(x[0]);
    ...
};
```

Formale Parameter können von Konstante initialisiert werden. In diesem Fall ist initialisierender Wert ein Default-Wert. Parameter, die dem initialisierten Parameter folgen, müssen auch initialisiert werden.

### Beispiel:

```
void func(int x, double y = 0.0, bool z = true)
{
    ...
};
```

Beim Aufruf solcher Funktion können initialisierte Parameter ausgelassen werden. Statt ihnen werden Default-Werte eingesetzt.

### Beispiel:

```
func(123, 0.5);
```

Parameter einfacher Typen werden nach der Bedeutung übertragen, d.h. Änderungen der entsprechenden lokalen Variable dieses Typs innerhalb aufgerufener Funktion nicht in aufrufender Funktion spiegeln wird. Felder jeder Typen und Daten des Typs Strukturen werden immer mit Link übertragen. Wenn es notwendig ist, Veränderung des Feldes oder Strukturinhalt zu verbieten, müssen Parameter dieser Typen mit Schlüsselwort `const` erklärt werden.

Es ist möglich, Parameter einfacher Typen durch Referenz zu übertragen. In diesem Fall ist Modifikation solcher Parameter entsprechende Variablen in aufgerufener Funktion beeinflusst, die durch Referenz übertragen wurden. Für Andeutung, dass der Parameter durch Referenz übertragen wird, muss nach dem Datentyp Modifikator `&` gestellt werden.

### Beispiel:

```
void func(int& x, double& y, double & z[])
{
    double calculated_tp;
    ...
}
```

```
for(int i=0; i<OrdersTotal(); i++)
{
    if(i==ArraySize(z)) break;
    if(OrderSelect(i)==false) break;
    z[i]=OrderOpenPrice();
};
x=i;
y=calculated_tp;
;}
```

Parameter die durch Referenz übertragen werden, können nicht mit Default-Werten initialisiert werden.

In Funktion können nicht mehr als 64 Parameter übertragen werden.

#### Sehen Sie auch

[Input Variablen](#), [Datentypen](#), [Inkapsulation und Erweiterungsfaehigkeit der Typen](#), [Initialisierung der Variablen](#), [Sichtbereich und Lebensdauer der Variablen](#), [Erzeugung und Entfernung der Objekte](#)

## Statische Variablen

Speicherklasse **static** bezeichnet statistische Variable. Modifikator static wird vor dem Datentyp angegeben.

**Beispiel:**

```
int somefunc()
{
    static int flag=10;
    ...
    return(flag);
; }
```

Statische Variable kann durch die ihrem Typ entsprechende Konstante oder Konstantausdruck initialisiert werden, zum Unterschied von einfacher lokalen Variable, die von jedem Ausdruck initialisiert werden kann.

Statische Variablen existieren seit Programmdurchführung und werden einmalig vor dem Aufruf spezialisierter Funktion OnInit() initialisiert. Wenn Anfangswerte nicht angegeben werden, nehmen Variablen der statistischen Klasse Nullwerte an.

Lokale Variablen, erklärt mit Schlüsselwort static bewahren ihre Werte innerhalb der ganzen Lebensdauer der Funktion. Beim der Funktion haben lokale Variablen die Werte, die sie beim vorletzten Aufruf hatten.

Alle Variablen im Block, ausser formaler Parameter der Funktion können als statisch bezeichnet werden.

**Beispiel:**

```
int Counter()
{
    static int count;
    count++;
    if(count%100==0) Print("Funktion Counter ist schon aufgerufen ",count," mal");
    return count;
; }

void OnStart()
{
    //---
    int c=345;
    for(int i=0;i<1000;i++)
    {
        int c=Counter();
    ; }
    Print("c =",c);
; }
```

**Sehen Sie auch**

Datentypen, Inkapsulation und Erweiterungsfaehigkeit der Typen, Initialisierung der Variablen,

Sichtbereich und Lebensdauer der Variablen, Erzeugung und Entfernung der Objekte, Statische Klassenmitglieder

## Globale Variablen

Globale Variablen werden durch Unterbringung ihrer Erklärungen ausser beschreibung bestimmter Funktion erzeugt. Globale Variablen werden auf demselben Niveau, wie Funktionen bestimmt, d.h. sie sind nicht lokal in einem Block.

### Beispiel:

```
int GlobalFlag=10;    // globale Variable
int OnStart()
{
    ...
};
```

Sichtbereich globaler Variablen - das ganze Programm, globale Variablen werden von allen Funktionen zugaenglich, bestimmt im Programm. Werden von Null initialisiert, wenn explizit anderer Anfangswert nicht zugeordnet wird. Globale Variable kann nur von der entsprechenden ihrem Typ Kinstante oder Konstantausdruck initialisiert werden.

Global variables are initialized only once after the program is loaded into the client terminal memory and before the first handling of the [Init](#) event. For global variables representing class objects, during their initialization the corresponding constructors are called. In scripts global variables are initialized before handling the [Start](#) event.

**Bemerkung:** Man soll nicht Variablen, erklärt auf dem flobalen Niveau mit globalen Variablen des kundenterminals verwechseln, der Zugang zu denen mittels der Funktionen [GlobalVariable...\(\)](#) durchgeführt wird.

### Sehen Sie auch

[Datentypen](#), [Inkapsulation und Erweiterungsfaehigkeit der Typen](#), [Initialisierung der Variablen](#), [Sichtbereich und Lebensdauer der Variablen](#), [Erzeugung und Entfernung der Objekte](#)



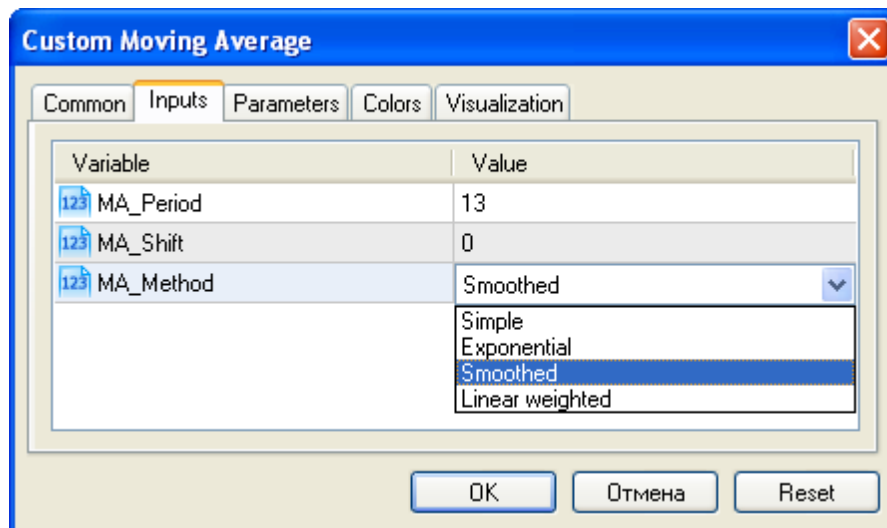
## Input Variablen

Speicherklasse `input` bestimmt die Aussenvariable. Modifikator `input` wird vor dem Datentyp angegeben. Den Wert von Variable mit dem Modifikator `input` innerhalb mql5-Programm darf man nicht verändern, diese Variable sind nur für Lesen zugänglich. Den Wert von input-Variablen kann der Benutzer vom Fenster der Programmeigenschaften.

### Beispiel:

```
//--- input parameters
input int      MA_Period=13;
input int      MA_Shift=0;
input ENUM_MA_METHOD MA_Method=MODE_SMA;
```

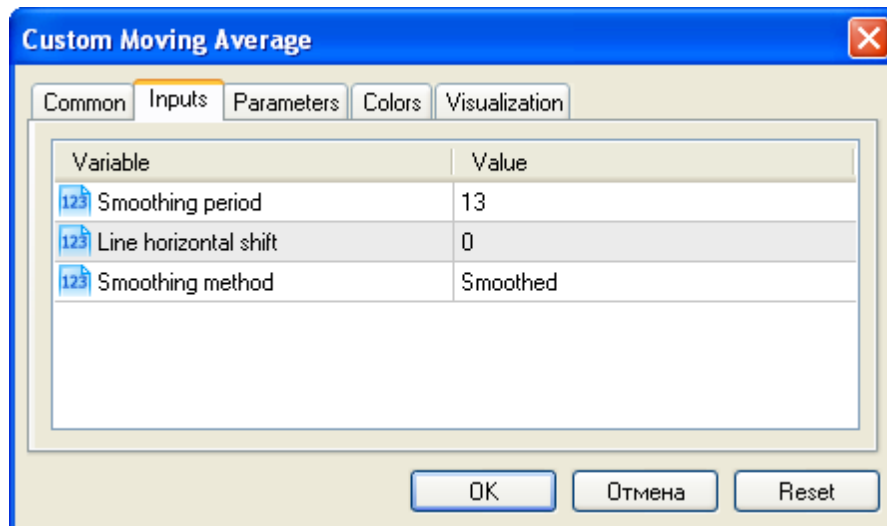
Input Variablen bestimmen Eingabeparameter des Programms, sie sind vom Fenster der Programmeigenschaften zugänglich.



Es ist möglich, ein anderes Verfahren, Eingabeparameter im Registerblatt "Inputs" darzustellen. Dafür wird einen Zeilenkommentar verwendet, der sich nach der Beschreibung des Eingabeparameters in der selben Zeile befinden muss. So, können für Eingabeparameter die für den Benutzer verständlichere Namen gegeben werden.

### Beispiel:

```
//--- input parameters
input int      InpMAPeriod=13;           // Smoothing period
input int      InpMAShift=0;             // Line horizontal shift
input ENUM_MA_METHOD InpMAMethod=MODE_SMA; // Smoothing method
```



**Bemerkung:** Felder und Variablen [zusammengesetzter Typen](#) können nicht als input-Variablen auftreten.

## Parameterübergabe beim Aufruf der Benutzerindikatoren aus mql5-Programmen

Benutzerindikatoren werden mittels der Funktion [iCustom\(\)](#) aufgerufen. Dabei müssen nach dem Namen des Benutzerindikators Parameter folgen in strenger Übereinstimmung mit Erklärung der input-Variablen des vorgegebenen Benutzerindikators. Wenn die Anzahl der angegebenen Parameter kleiner ist als die Anzahl der erklärten input-Variablen im aufgerufenen Benutzerindikator, werden die fehlenden Parameter von Werten ausgefüllt, die bei der Variablenerklärung angegeben wurden.

Wenn im Benutzerindikator die Funktion [OnCalculate](#) erster Art (d.h. der Indikator ist auf einem Datenfeld) verwendet wird, muss als letzter Parameter beim Aufruf solches Benutzerindikators einer der Werte [ENUM\\_APPLIED\\_PRICE](#) oder handle eines anderen Indikators auftreten. Dabei müssen alle Parameter, die input-Variablen entsprechen, explizit angegeben werden.

## Aufzählungen als input-Parameter

Als input-Variablen (Eingabeparameter für mql5-Programme) können nicht nur die von der Sprache MQL5 vorausgesehene eingebettete Aufzählungen verwendet werden, sondern auch die vom Benutzer vorbestimmten Aufzählungen. Z.B. können wir die Aufzählung `dayOfWeek` erzeugen, die Wochentage beschreibt und input-Variable für Angeben konkretes Wochentages nicht als Ziffer, sondern in der gewöhnlicheren für Benutzer Art verwenden.

**Beispiel:**

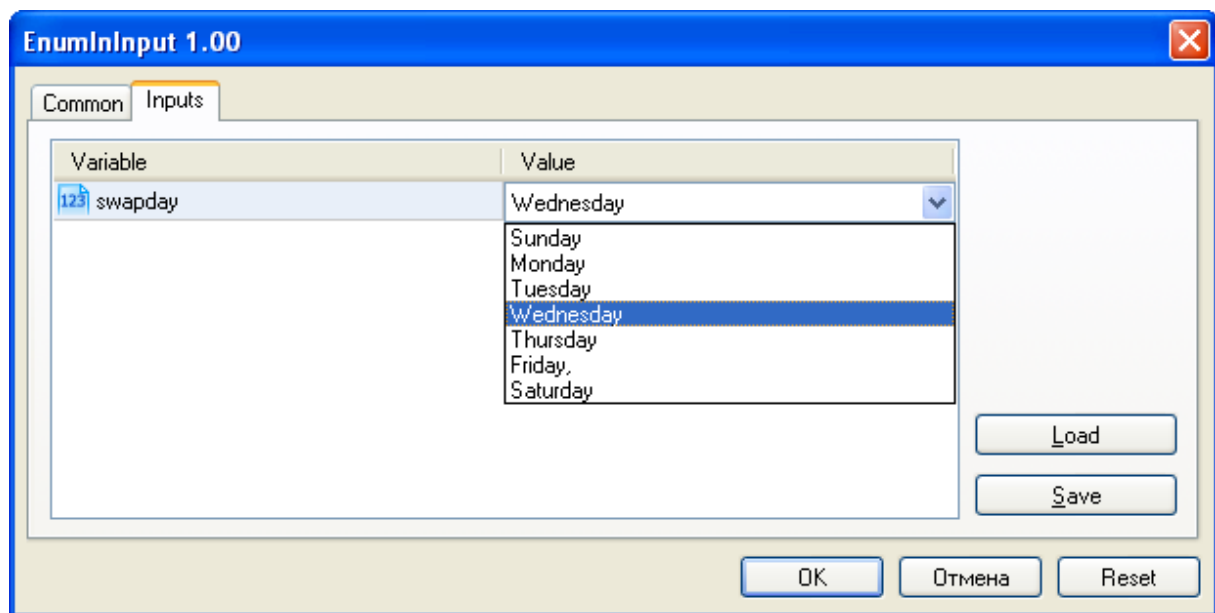
```
#property script_show_inputs
//--- day of week
enum dayOfWeek
{
    S=0,    // Sunday
    M=1,    // Monday
    T=2,    // Tuesday
    W=3,    // Wednesday
}
```

```

Th=4,    // Thursday
Fr=5,    // Friday,
St=6,    // Saturday
};
//--- input parameters
input dayOfWeek swapday=W;

```

Damit der Benutzer beim Scriptstart den nötigen Wert aus Eigenschaftsfenster wählen kann, verwenden wir Preprozessorbefehl `#property script_show_inputs`. Dann aktivieren wir Script und können aus der Liste einen der Aufzählungswerte `dayOfWeek` wählen. Dann aktivieren wir Script `EnumInput` und übergehen zum Registerblatt "Parameter". Ein Default-Wert des Parameters `swapday` ist Mittwoch (W=3), aber wir können einen anderen Wert zuordnen und diesen Wert für Änderung der Programmarbeit verwenden.



Zahl der möglichen Aufzählungswerte ist begrenzt. Darum wird Auswahl des Eingabewertes Auswahlliste verwendet. Als Werte gezeigt in der Liste werden mnemonische Namen der Aufzählungsglieder verwendet. Wenn dem mnemonischen Namen Kommentar gegenübergestellt wird, wie es in unserem Beispiel gezeigt ist, wird anstatt des mnemonischen Namens Kommentarinhalt verwendet.

Jeder Wert aus der Aufzählung `dayOfWeek` hat seinen Wert von 0 bis 6, aber in der Parameterliste werden Kommentare gezeigt angegeben für jeden Wert. Das verleiht zusätzliche Anpassungsfähigkeit für Programmschreiben mit vollständigen Beschreibungen der Eingabeparameter.

## Variablen mit dem Modifikator `input`

Variablen mit dem Modifikator `input` erlauben nicht nur die Werte der externen Parameter bei dem Start der Programme anzugeben, sondern auch eine grosse Rolle bei der Optimierung der Handelsstrategien im Tester spielen. Jede im Expert Advisor erklärte `input`-Variable, mit Ausnahme des Typs `string`, kann an der Optimierung teilnehmen.

In einigen Fällen ist es notwendig, einige externe Parameter des Programms von der Erstellung des Bereichs von allen möglichen Durchgängen im Tester auszuschließen. Speziell für solche Fälle gibt es einen Modifikator von Speicher `sinput`. `sinput` ist eine Kurzform der Erklärung von statischer globaler Variable: `sinput = static input`. Es bedeutet die folgende Erklärung im Code des Expert-Advisors

```
sinput      int layers=6;    // Number of layers
```

wird gleichbedeutend mit der vollständigen Erklärung sein

```
static input int layers=6;    // Number of layers
```

Die mit dem Modifikator `sinput` erklärte Variable ist ein Eingabeparameter des MQL5-programms, der Wert dieses Parameters kann bei seinem Start geändert werden. Dabei nimmt diese Variable nicht an dem Optimierungsprozess der Eingabeparameter teil, das heißt werden ihre Werte bei der Suche des besten Satzes der Parameter nach dem angegebenen Kriterium nicht durchgesucht.

| Variable                                                          | Value | Start | Step | Stop | Steps  |
|-------------------------------------------------------------------|-------|-------|------|------|--------|
| <input type="checkbox"/> Number of layers                         | 6     |       |      |      |        |
| <input checked="" type="checkbox"/> Neurons in a layer            | 30    | 30    | 1    | 300  | 271    |
| <input checked="" type="checkbox"/> Number of bars to be analyzed | 13    | 13    | 1    | 130  | 118    |
| <input checked="" type="checkbox"/> Forecast horizon              | 2     | 2     | 1    | 20   | 19     |
| <input type="checkbox"/> Network type                             | 0     | 0     | 1    | 10   |        |
|                                                                   |       |       |      |      | 607582 |

Settings Inputs Agents Journal

For Help, press F1 Default

Die Abbildung zeigt, dass der Expert-Advisor fünf externe Parameter hat. Der Parameter "Anzahl von Ebenen" wird als `sinput` erklärt und ist gleich 6. Dieser Parameter kann nicht während einer Optimierung der Handelsstrategie geändert werden. Man kann für ihn den notwendigen Wert angeben, der weiter verwendet wird. Felder für Start, Stop und Schritt für diese Variable sind für die Einstellung der Werte nicht verfügbar.

Nachdem wir den Modifikator `sinput` für Variable angegeben haben, verbieten wir dem Benutzer diesen Parameter zu optimieren. Es bedeutet, dass der Terminal Benutzer für sie die Anfangs- und Endwerte für automatisches Durchsuchen in einem bestimmten Bereich bei der Optimierung im Strategie-Tester nicht angeben kann.

Aber dabei gibt es eine Ausnahme von dieser Regel - `sinput`-Variablen kann man in Optimierungsaufgaben mit Hilfe der Funktion [ParameterSetRange\(\)](#) variieren. Diese Funktion wurde für die Programmsteuerung des Raums der verfügbaren Werte für jede `input`-Variable speziell erstellt, einschließlich jener, die für `static input` (`sinput`) erklärt sind. Eine weitere Funktion [ParameterGetRange\(\)](#) erlaubt es Ihnen (im Handler [OnTesterInit\(\)](#)) die Werte der `input`-Variablen zu erhalten, wenn Optimierung gestartet wird, und falls notwendig, den Schritt der Veränderung und den Bereich neu anzugeben, innerhalb dessen der Wert des optimierten Parameters durchgesucht wird.

Daher erlaubt die Kombination von `sinput` Modifikator mit zwei Funktionen für die Arbeit mit `input`-Parametern die flexiblen Regeln für die Einstellung der Optimierungsintervallen von einigen `input`-Variablen je nach der Werten der anderen `input`-Variablen zu schaffen.

Sehen Sie auch

[iCustom](#), [Aufzählungen](#), [Programmeigenschaften](#)

## Extern Variablen

Schlüsselwort `extern` wird verwendet, um Identifikatoren der Variablen als Identifikatoren [statischer Speicherklasse](#) mit globaler [Lebensdauer](#) zu erklären. Solche Variable existieren seit Anfang der Programmdurchführung und für sie wird Speicher sofort nach Anfang der Programmdurchführung verteilt und initialisiert.

Man kann Programme erzeugen, die aus mehreren Ausgangsdateien bestehen. Dafür wird die Anweisung des Preprozessors `#include` verwendet. Variablen, die als `extern` erklärt werden mit demselben Typ und Identifikator, können in verschiedenen Ausgangsdateien eines Projekts existieren.

Bei Compilierung des ganzen Projektes werden alle `extern`-Variablen, die denselben Typ und Identifikator haben, werden mit einem. `Extern`-Variablen sind für getrennte Compilierung der Ausgangsdateien nützlich. `Extern`-Variablen können initialisiert werden, aber nur einmal - es ist nicht zulaessig, dass mehrere initialisierte `extern`-Variablen eines Typs und mit demselben Identifikator existieren.

### Sehen Sie auch

[Datentypen](#), [Inkapsulation und Erweiterungsfaehigkeit der Typen](#), [Initialisierung der Variablen](#), [Sichtbereich und Lebensdauer der Variablen](#), [Erzeugung und Entfernung der Objekte](#)

## Initialisierung der Variablen

Jede Variable kann bei der Definition initialisiert werden. Wenn explizite Initialisierung der Variable nicht durchgeführt wird, kann der Wert, gespeichert in der Variable, ganz verschieden sein. Implizite Initialisierung wird nicht durchgeführt.

Globale und statische Variablen können nur von Konstante des entsprechenden Typs oder Konstantausdruck initialisiert werden. Lokale Variablen können von jedem Ausdruck initialisiert werden, nicht nur von der Konstante.

Initialisierung von globalen und statischen Variablen wird einmalig durchgeführt. Initialisierung lokaler Variablen wird jedesmal beim Aufruf entsprechender Funktionen durchgeführt.

### Beispiele:

```
int    n        = 1;
string s        = "hello";
double f[]      = { 0.0, 0.236, 0.382, 0.5, 0.618, 1.0 };
int    a[4][4]  = { {1, 1, 1, 1}, {2, 2, 2, 2}, {3, 3, 3, 3}, {4, 4, 4, 4} };
//--- aus Tetris
int    right[4]={WIDTH_IN_PIXELS+VERT_BORDER,WIDTH_IN_PIXELS+VERT_BORDER,
                WIDTH_IN_PIXELS+VERT_BORDER,WIDTH_IN_PIXELS+VERT_BORDER};
//--- Initialisierung aller Felder in der Struktur mit der NULL-Werte
MqlTradeRequest request={0};
```

Liste von Werten der Feldelemente muss in geschweiften Klammern stehen. Ausgelassene initialisierende Folgen gelten als die der Null gleichen. In der initialisierenden Folge muss mindestens ein Wert geben: von diesem Wert wird das erste Element entsprechender Struktur oder des Feldes initialisiert, fehlende Elemente gelten als die der Null gleichen.

Wenn die Größe des zu initialisierten Feldes nicht angegeben wird, wird die vom Compiler festgestellt, ausgehend von der gröesse der initialisierenden Folge. Vieldimensionale Felder können nicht von eindimensionaler Folge (Folge ohne zusätzliche geschweifte Klammern) initialisiert werden, ausser einem Fall - wenn nur ein initialisiertes Element angegeben wird (in der Regel Null).

Felder (darunter auch auf dem lokalen Niveau erklärt) können nur von Konstanten initialisiert werden.

### Beispiele:

```
struct str3
{
    int          low_part;
    int          high_part;
};
struct str10
{
    str3         s3;
    double       d1[10];
    int          i3;
};
void OnStart()
{
```

```
str10 s10_1={{1,0},{1.0,2.1,3.2,4.4,5.3,6.1,7.8,8.7,9.2,10.0},100};
str10 s10_2={{1,0},{0},100};
str10 s10_3={{1,0},{1.0}};
//---
Print("1.  s10_1.d1[5] = ",s10_1.d1[5]);
Print("2.  s10_2.d1[5] = ",s10_2.d1[5]);
Print("3.  s10_3.d1[5] = ",s10_3.d1[5]);
Print("4.  s10_3.d1[0] = ",s10_3.d1[0]);
; }
```

Für Variablen des Typs Strukturen ist teilweise Initialisierung erlaubt, gilt dies auch für statische Arrays (mit einer expliziten Größe). Sie können eine oder mehrere der ersten Elemente einer Struktur oder Array initialisieren, werden die restlichen Elemente in diesem Fall mit Nullen initialisiert werden.

Sehen Sie auch

[Datentypen](#), [Inkapsulation und Erweiterungsfaehigkeit der Typen](#), [Sichtbereich und Lebensdauer der Variablen](#), [Erzeugung und Entfernung der Objekte](#)

## Sichtbereich und Lebensdauer der Variablen

Es gibt zwei Typen von Sichtbereich: lokaler Sichtbereich und globaler Sichtbereich.

Variable, die ausser aller Funktionen erklärt wurde, wird in globalen Sichtbereich eingestellt. Zugang zu diesen Variablen kann aus jedem Programmplatz erfolgen. Solche Variablen befinden sich im globalen Pool des Speichers, darum faellt ihre Lebensdauer mit der Programmlebensdauer.

Variable, erklärt innerhalb des Blocks (Codeteil in geschweiften Klammern) gehört dem lokalen Sichtbereich. Solche Variable ist unsichtbar (darum unzugänglich) ausser Block, in dem sie erklärt wird. Die verbreitetste lokale Erklärung - Variable, erklärt innerhalb der Funktion. Variable, die lokal erklärt wird befindet sich im Stoss und ihre Lebensdauer faellt mit der Lebensdauer der Funktion zusammen.

Da Sichtbereich der lokalen Variable Block, in dem sie erklärt wurde, ist, gibt es die Möglichkeit, Variablen mit Namen zu erklären, der mit den Variablennamen, erklärt in anderen Blocks, und auch erklärt auf den hoeheren Niveaus bis zum globalen, zusammenfaellt.

**Beispiel:**

```
void CalculateLWMA(int rates_total,int prev_calculated,int begin,const double &price[])
{
    int          i,limit;
    static int weightsum=0;
    double        sum=0;
    //---
    if(prev_calculated==0)
    {
        limit=MA_Period+begin;
        //--- set empty value for first limit bars
        for(i=0; i<limit; i++) LineBuffer[i]=0.0;
        //--- calculate first visible value
        double firstValue=0;
        for(int i=begin; i<limit; i++)
        {
            int k=i-begin+1;
            weightsum+=k;
            firstValue+=k*price[i];
        }
        firstValue/=(double)weightsum;
        LineBuffer[limit-1]=firstValue;
    }
    else
    {
        limit=prev_calculated-1;
    }

    for(i=limit; i<rates_total; i++)
    {
        sum=0;
```



```
for(int j=0; j<MA_Period; j++) sum+=(MA_Period-j)*price[i-j];
LineBuffer[i]=sum/weightsum;
; }
//---
; }
```

Legen Sie Wert auf Variable i, erklärt in der Zeile:

```
for(int i=begin; i<limit; i++)
{
    int k=i-begin+1;
    weightsum+=k;
    firstValue+=k*price[i];
; }
```

Ihr Sichtbereich ist ausschliesslich der Zyklus for, ausser dieses Zyklus fungiert eine andere Variable mit demselben Namen, erklärt am Anfang der Funktion. Außerdem wird im Zykluskoerper die Variable k erklärt, deren Sichtbereich Zykluskoerper ist.

Lokale Variablen können mit dem Zugangsspezifikator [static](#) erklärt werden. In diesem Fall stellt der Compiler diese Variable in globalenSpeicherpool. Darum faellt Lebensdauer statischer Variabler mit der Lebensdauer der Programmlebensdauer zusammen. Dabei wird Sichtbereich dieser Variable vom Block, in dem sie erklärt wurde, begrenzt.

Sehen Sie auch

[Datentypen](#), [Inkapsulation und Erweiterungsfaehigkeit der Typen](#), [Initialisierung der Variablen](#), [Erzeugung und Entfernung der Objekte](#)

## Erzeugung und Entfernung der Objekte

Nach der Ladung des mql5-Programms wird für jede Variable Speicher entsprechend dem Typ der Variable verteilt. Variablen werden in zwei Typen nach dem Zugang aufgeteilt - [globale Variablen](#) und [lokale Variablen](#), und nach den Speicherklassen: [Eingabeparameter](#) des mql5-Programms, [statische](#) und automatische. Jede Variable wird wenn notwendig durch den entsprechenden Wert [initialisiert](#). Nach der Verwendung wird die Variable deinitialisiert und der von ihr verwendete Speicher kehrt zum ausführenden System MQL5.

## Initialisierung und Deinitialisierung der globalen Variablen

Initialisierung der globalen Variablen erfolgt automatisch nach dem Laden des mql5-Programms und vor dem Aufruf jeder Funktion. Bei der Initialisierung erfolgt automatische Zuordnung der Initialwerte den Variablen [einfacher](#) Typen und wird der Konstrukteur für Objekte aufgerufen (wenn er vorhanden ist). [Eingabeparameter](#) werden immer auf dem globalen Niveau erklärt, durch die Werte initialisiert, die von Benutzern im Dialog beim Ausführungslauf des mql5-Programms.

Abgesehen davon, dass [statische](#) Variablen gewöhnlich auf dem lokalen Niveau erklärt werden, wird der Speicher für diese Variablen im Voraus verteilt und Initialisierung erfolgt sofort nach der Ladung des Programms, wie auch für [globale](#) Variablen.

Ordnung der Initialisierung erfolgt der Ordnung der Erklärung der Variable im Programm und Deinitialisierung erfolgt umgekehrt vor der Ausladung des mql5-Programms. Diese Regel gilt nur für diejenigen Variablen, die nicht von der Anweisung `new` erzeugt wurden. Solche Variablen werden sofort nach der Ladung erzeugt und initialisiert, und unmittelbar vor der Ausladung werden sie deinitialisiert.

## Initialisierung und Deinitialisierung der lokalen Variablen

Wenn die Variable, die auf dem lokalen Niveau erklärt wird, nicht statisch ist, erfolgt die Speicherverteilung für eine solche Variable automatisch. Lokale Variablen, wie auch globale werden automatisch in dem Augenblick initialisiert, wenn während der Ausführung des Programms Erklärung der lokalen Variable erfolgt. So entspricht die Ordnung der Initialisierung der Ordnung der Erklärung.

Lokale Variablen werden am Ende des Programmblocks deinitialisiert, in dem sie erklärt wurden, und in der Reihenfolge, die ihrer Erklärung umgekehrt ist. Block eines Programms ist eine [zusammengesetzte Anweisung](#), die ein Teil der Auswahlanweisung [switch](#), Zykusanweisung ([for](#), [while](#), [do-while](#)), [Körper der Funktion](#) oder Teil der [Anweisung if-else](#) sein kann.

Initialisierung der lokalen Variablen erfolgt nur in dem Moment, wenn während der Initialisierung des Programms Erklärung der Variable erfolgt. Wenn während der Programmausführung das Block, in dem die Variable erklärt wurde, nicht ausgeführt wurde, wird eine solche Variable nicht initialisiert.

## Initialisierung und Deinitialisierung der dynamisch verteilten Objekte

Ein Besonderer Fall stellen [Objektanzeiger](#) dar, denn Erklärung des Anzeigers führt nicht zur Initialisierung des entsprechenden Objektes. Die dynamisch verteilten Objekte werden nur im Augenblick der Erzeugung des Klassenexemplares durch die [Anweisung new](#). Initialisierung des Objektes setzt Aufruf des Konstruktors einer entsprechenden Klasse voraus. Wenn es keinen

entsprechenden Konstruktor in der Klasse gibt, werden seine Glieder des [einfachen Typs](#), automatisch nicht initialisiert; Glieder der Typen [Zeile](#), [dynamisches Feld](#) und [zusammengesetztes Objekt](#) werden automatisch initialisiert.

Anzeiger können auf dem globalen und auf dem lokalen Niveau erklärt werden, und dabei durch einen leeren Wert [NULL](#) oder durch den Wert eines Anzeiger desselben oder des [heritierten](#) Typs initialisiert werden können. Wenn für den Anzeiger, der auf dem lokalen Niveau erklärt wurde, die Anweisung *new* aufgerufen wird, muss auch die Anweisung *delete* für diesen Anzeiger vor dem Ausgang vom diesen Niveau ausgeführt. Anderenfalls wird der Anzeiger verlorengehen und das Objekt kann nicht explizit entfernt werden.

Alle Objekte, die mittels des Ausdrucks *Anzeiger\_des Objektes=new Name\_der Klasse* erzeugt wurden, müssen im nachhinein mittels der Anweisung *delete(Anzeiger\_des Objektes)* entfernt werden. Wenn eine Variable dieser Art nach dem Beenden des Programms aus irgendwelchen Gründen von der [Anweisung delete](#) nicht entfernt wurde, wird darüber eine Meldung im Journal "Experten" ausgegeben. Man kann mehrere Variablen erklären und allen den Anzeiger eines Objektes zuordnen.

Wenn ein dynamisch erzeugtes Objekt Konstruktor hat, wird dieser Konstruktor aufgerufen werden während die Anweisung *new* ausgeführt wird. Wenn Objekt einen Destructor hat, wird er aufgerufen werden, während die Anweisung *delete* ausgeführt wird.

So werden dynamisch verteilte Objekte nur während der Ausführung durch die Anweisung *new* erzeugt und werden bestimmt entweder von der Anweisung *delete* oder vom automatisch ausführenden System MQL5 wenn das Programm ausgeladen wird. Ordnung der Erklärung der Anzeiger der dynamisch erzeugten Objekte beeinflusst die Ordnung ihrer Initialisierung nicht. Die Ordnung der Initialisierung und die Ordnung der Deinitialisierung werden vollständig vom Programmierer kontrolliert.

## Arbeit mit dynamischen Speicher

Beim Arbeiten mit dynamischen Arrays wird der freigegebene Speicher unmittelbar an das System zurückgegeben.

Bei der Erstellung eines dynamischen Klassenobjekt durch [new](#), wird der Speicher zuerst im Pool von Klassenspeicher, mit dem der Speicher-Manager arbeitet, gesucht, und wenn der Pool nicht genug Speicher hat, wird der Speicher im System angefordert. Wenn Sie ein dynamisches Objekt durch [delete](#) löschen, wird Speicher, der vom Objekt besetzt ist, in den Pool von Klassenspeicher zurückgegeben.

Der Speicher-Manager gibt den Speicher an das System zurück sofort nach der Ausgang aus der folgenden Event-Handler-Funktionen: [OnInit\(\)](#), [OnDeinit\(\)](#), [OnStart\(\)](#), [OnTick\(\)](#), [OnCalculate\(\)](#), [OnTimer\(\)](#), [OnTrade\(\)](#), [OnTester\(\)](#), [OnTesterInit\(\)](#), [OnTesterPass\(\)](#), [OnTesterDeinit\(\)](#), [OnChartEvent\(\)](#), [OnBookEvent\(\)](#).

## Kurzcharakteristik der Variablen

Hauptinformation über Erzeugung, Entfernung, Aufruf der Konstruktors und Destruktors werden in der Kurztabelle angegeben.

|  | Globale automatische Variable | Lokale automatische Variable | Dynamisch erzeugtes Objekt |
|--|-------------------------------|------------------------------|----------------------------|
|  |                               |                              |                            |

|                                      |                                                                                                    |                                                                                                    |                                                                                             |
|--------------------------------------|----------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------|
| <b>Initialisierung</b>               | sofort nach der Ladung des mql5-Programms                                                          | Beim Erreichen des Codes der Zeile, wo sie erklärt ist                                             | bei Ausführung der Anweisung <b>new</b>                                                     |
| <b>Ordnung der Initialisierung</b>   | in Form von Erklärung                                                                              | in Form von Erklärung                                                                              | unabhängig von der Ordnung der Erklärung                                                    |
| <b>Deinitialisierung</b>             | vor der Ausladung des mql5-Programms                                                               | beim Ausgang der Ausführung aus dem Block der Erklärung                                            | bei Ausführung der Anweisung <b>delete</b> oder vor Ausladung des mql5-Programms            |
| <b>Ordnung der Deinitialisierung</b> | in der Reihenfolge, die der Initialisierung umgekehrt ist                                          | in der Reihenfolge, die der Initialisierung umgekehrt ist                                          | nicht abhängig von der Ordnung der Initialisierung                                          |
| <b>Aufruf des Konstruktors</b>       | bei Ladung des mql5-Programms                                                                      | bei der Initialisierung                                                                            | bei Ausführung der Anweisung <b>new</b>                                                     |
| <b>Aufruf des Destruktors</b>        | bei Ausladung des mql5-Programms                                                                   | beim Ausgang aus dem Block, in dem die Variable initialisiert wurde                                | bei Ausführung der Anweisung <b>delete</b>                                                  |
| <b>Meldung über Fehler</b>           | Meldung im Journal "Experten" über den Versuch der Entfernung eines automatisch erzeugten Objektes | Meldung im Journal "Experten" über den Versuch der Entfernung eines automatisch erzeugten Objektes | Meldung im Journal "Experten" über nicht entfernte Objekte bei Ausladung des mql5-Programms |

Sehen Sie auch

[Datentypen](#), [Inkapsulation und Erweiterungsfaehigkeit der Typen](#), [Initialisierung der Variablen](#), [Sichtbereich und Lebensdauer der Variablen](#), [Erzeugung und Entfernung der Objekte](#)

## Preprozessor

Preprozessor - ein Sondersubsystem des Compilers MQL5, das den Ausgangstext des Programms unmittelbar vor seiner Compilierung vorbereitet. Ausgangsprogrammtext vor der Compilierung vorbereitet.

Preprozessor ermöglicht, die Lesbarkeit des Ausgangskodes zu verbessern. Strukturierung des Codes kann durch Eintragung einzelner Dateien mit Ausgangskoden der mql5-Programme erreicht werden. Der Verbesserung von Kodelesbarkeit befoerdert auch die Moeglichkeit, mnemonische Namen den einzelnen Konstanten zuzuordnen.

Preprozessor ermöglicht auch, spezifische Parameter der mql5-Programme zu bestimmen:

- [Konstanten erklären](#)
- [Programmeigenschaften einstellen](#)
- [Dateien in Programmtext einschliessen](#)
- [Funktionen importieren](#)
- [Die bedingte Kompilierung verwenden](#)

Wenn als erstes Symbol in Programmzeile das Symbol #, verwendet wird, ist diese Zeile befehlende Anweisung des Preprozessors. Befehlende Anweisung des Preprozessors beendet mit dem Zeilenvorschubszeichen.

## Makrosubstitution (#define, #undef)

Befehlende Anweisung `#define` kann verwendet werden, um mnemonische Namen den zuzuordnen. Es gibt zwei Formen:

```
#define identifier expression           // parameterfreie Form
#define identifier(par1,... par8) expression // parametrische Form
```

Befehlsanweisung `#define` ersetzt **expression** anstatt aller folgenden gefundenen Eintragungen von `identifier` im Ausgangstext. `identifier` wird erst dann ersetzt, wenn er einzelner token darstellt. `identifier` wird nicht ersetzt, wenn er ein Teil des Kommentars, der Zeile oder eines anderen laengeren Identifikators ist.

für den Identifikator der Konstante gelten dieselbe Regeln, wie für die Konstantennamen. Der wert kann jeder Art sein:

```
#define ABC          100
#define PI           3.14
#define COMPANY_NAME "MetaQuotes Software Corp."
...
void ShowCopyright()
{
    Print("Copyright 2001-2009, ",COMPANY_NAME);
    Print("http://www.metaquotes.net");
};
```

**expression** kann aus mehreren tokens bestehen, wie Schluesselwoerter, Konstanten, Konstant- und Nichtkonstantausdrücke. **expression** beendetmit dem Zeilenende und kann nicht auf die naechste Zeile übertragen werden.

### Beispiel:

```
#define TWO          2
#define THREE        3
#define INCOMPLETE TWO+THREE
#define COMPLETE (TWO+THREE)
void OnStart()
{
    Print("2 + 3*2 = ",INCOMPLETE*2);
    Print("(2 + 3)*2 = ",COMPLETE*2);
};
/* Ergebnis
2+3*2 = 8
(2+3)*2 = 10
*/
```

### Parametrische Form #define

Mit der parametrischen Form, werden alle folgenden gefunden Einträge von `identifier` werden durch `expression` auf der Grundlage der aktuellen Parameter ersetzt werden. Zum Beispiel,

```
// Beispiel mit zwei Parameter a und b
#define A 2+3
#define B 5-1
#define MUL(a, b) ((a)*(b))

double c=MUL(A,B);
Print("c=",c);
/*
Expression double c=MUL(A,B);
ist gleichbedeutend mit double c=((2+3)*(5-1));
*/
// Ergebnis
// c=20
```

Achten Sie darauf, dass Parameter in Klammern eingeschlossen sind, wenn Sie den Parameter in expression verwenden, da dies nicht-offensichtlichen Fehlern, die schwer zu finden sind, vermeidet. Wenn wir den Code neu schreiben, ohne die Klammern, wird das Ergebnis anders sein:

```
// Beispiel mit zwei Parameter a und b
#define A 2+3
#define B 5-1
#define MUL(a, b) a*b

double c=MUL(A,B);
Print("c=",c);
/*
Expression double c=MUL(A,B);
ist gleichbedeutend mit double c=2+3*5-1;
*/
// Ergebnis
// c=16
```

Bei Verwendung der parametrischen Form sind maximal 8 Parameter erlaubt.

```
// richtige parametrische Form
#define LOG(text) Print(__FILE__,"(",__LINE__,") :",text) // ein Parameter - 'text'

// falsche parametrische Form
#define WRONG_DEF(p1, p2, p3, p4, p5, p6, p7, p8, p9) p1+p2+p3+p4 // mehr als 8 Parameter
```

## Директива #undef

Директива #undef предназначена для отмены макроса, объявленного ранее.

Пример:

```
#define MACRO

void func1()
{
    #ifdef MACRO
        Print("MACRO is defined in",__FUNCTION__);
    #else
        Print("MACRO is not defined in",__FUNCTION__);
    #endif
}
```

```
#endif
}

#undef MACRO

void func2 ()
{
#ifdef MACRO
    Print("MACRO is defined in ", __FUNCTION__);
#else
    Print("MACRO is not defined in ", __FUNCTION__);
#endif
}

void OnStart ()
{
    func1 ();
    func2 ();
}

/* Результат:
MACRO is defined in func1
MACRO is not defined in func2
*/
```

#### Sehen Sie auch

[Identifikatoren](#), [Symbolkonstanten](#)



## Programmeigenschaften (#property)

Bei jedem mql5-Programm können zusätzliche spezifische Parameter *#property* angegeben werden, die dem Kundenterminal helfen, Programme richtig zu bedienen. In erster Linie betrifft es Außeneinstellungen der Identifikatoren. Eigenschaften, beschrieben in Schaltdateien, werden völlig ignoriert. Eigenschaften müssen in der Haupt mql5-Datei vorbestimmt werden.

```
#property Identifikator Wert
```

Compiler schreibt in Einstellungen des ausführenden Moduls erklärte Werte.

| Konstante   | Typ                    | Beschreibung                                                                                                                                                                                                                                                                                               |
|-------------|------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| icon        | <a href="#">string</a> | Pfad zur Datei mit dem Bild, das als ein Icon des EX5-Programms verwendet werden soll. Pfadangaberegeln sind die gleichen wie für die <a href="#">Ressourcen</a> . Die Eigenschaft muss in dem Hauptmodul mit dem MQL5 Quellcode angegeben werden. Die Icondatei muss im <a href="#">ICO</a> -Format sein. |
| link        | <a href="#">string</a> | Verweis auf die Site der Gesellschaft-Hersteller                                                                                                                                                                                                                                                           |
| copyright   | <a href="#">string</a> | Name der Gesellschaft-Hersteller                                                                                                                                                                                                                                                                           |
| version     | <a href="#">string</a> | Programmversion, nicht mehr als 31 Zeichen                                                                                                                                                                                                                                                                 |
| description | <a href="#">string</a> | kurze Textbeschreibung des mql5-Programms. können einige descriptions vorhanden sein, jede von denen eine Textzeile beschreibt. Allgemeine Länge aller descriptions kann nicht mehr als 511 Zeichen unter Beachtung von Zeilenvorschubszeichen                                                             |
| stacksize   | <a href="#">int</a>    | Указывает размер <a href="#">стека</a> для MQL5 программы, стек достаточного объема требуется в случае выполнения рекурсивных вызовов функций. При запуске скрипта или                                                                                                                                     |

|                           |                        |                                                                                                                                                                                                                                                 |
|---------------------------|------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                           |                        | <p>эксперта на графике выделяется стек не менее 8Мб, для индикаторов свойство не работает - стек всегда фиксированного объема в 1Мб.</p> <p>При запуске в тестере программе всегда выделяется стек в размере 16 Мб.</p>                         |
| library                   |                        | <p>Bibliothek; wird von keiner Startfunktion zugeordnet, Funktionen mit dem <a href="#">Modifikator export</a> können in andere mql5-Programme <a href="#">importiert</a> werden</p>                                                            |
| indicator_applied_price   | <a href="#">int</a>    | <p>gibt Default-Wert für Feld "<a href="#">Apply to</a>" vor. Kann einer der Aufzählungswerte <a href="#">ENUM_APPLIED_PRICE</a> vorgegeben werden. Wenn die Eigenschaft nicht vorgegeben wird, wird als Default-Wert PRICE_CLOSE verwendet</p> |
| indicator_chart_window    |                        | <p>Indikator ins Chartfenster ausgeben</p>                                                                                                                                                                                                      |
| indicator_separate_window |                        | <p>Indikator im selbststaendigen Fenster ausgeben</p>                                                                                                                                                                                           |
| indicator_height          | <a href="#">int</a>    | <p>Feste Höhe des Indikator-Subfensters in Pixel (Eigenschaft <a href="#">INDICATOR_HEIGHT</a>)</p>                                                                                                                                             |
| indicator_buffers         | <a href="#">int</a>    | <p>Anzahl der Puffer für Indikatorberechnung</p>                                                                                                                                                                                                |
| indicator_plots           | <a href="#">int</a>    | <p>Anzahl der <a href="#">graphischen Reihen</a> in Indikator</p>                                                                                                                                                                               |
| indicator_minimum         | <a href="#">double</a> | <p>untere Skalabegrenzung des selbststaendigen Fensters des Indikators</p>                                                                                                                                                                      |
| indicator_maximum         | <a href="#">double</a> | <p>obere Skalebegrenzung des selbststaendigen Fensters des Indikators</p>                                                                                                                                                                       |
| indicator_labelN          | <a href="#">string</a> | <p>gibt Marke für N- <a href="#">graphische Serie</a> vor, die im Fenster DataWindow dargestellt wird.</p>                                                                                                                                      |

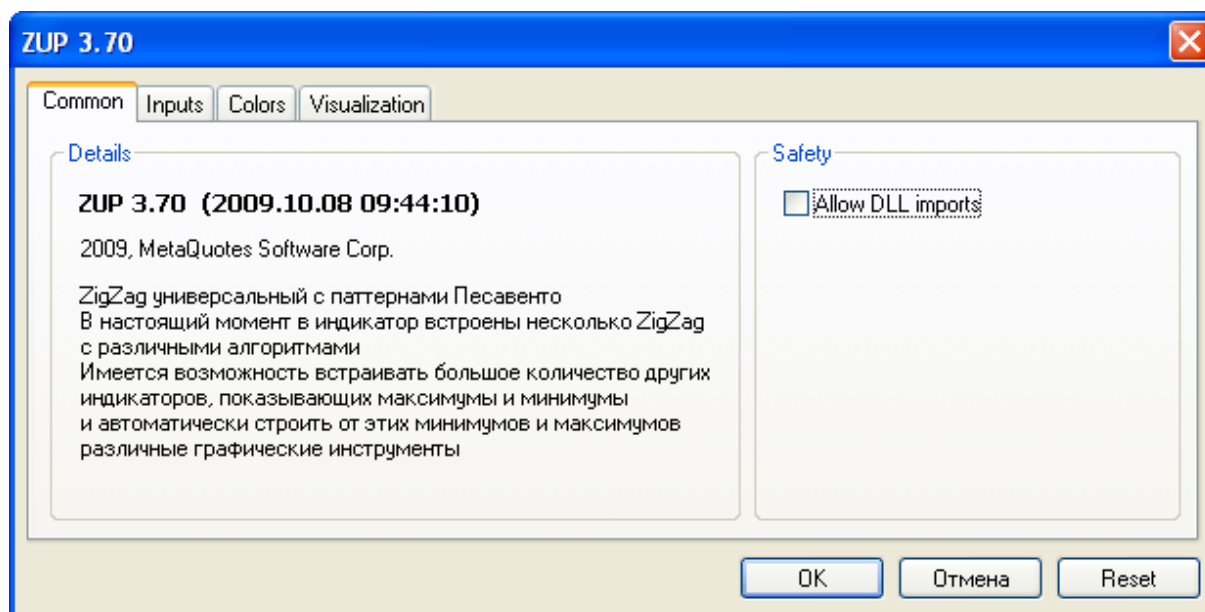
|                      |                        |                                                                                                                                                                                                 |
|----------------------|------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                      |                        | Für graphische Serien, die mehrere Indikatorpuffer erfordern (DRAW_CANDLES, DRAW_FILLING und andere), werden die Markennamen durch Begrenzer vorgegeben `;`.                                    |
| indicator_colorN     | <a href="#">color</a>  | Farbe für Linienausgabe N, wo N - Nummer einer <a href="#">graphischen Serie</a> ; Numerierung faengt mit 1 an                                                                                  |
| indicator_widthN     | <a href="#">int</a>    | Linienbreite in der <a href="#">graphischen Serie</a> , wo N - Nummer der graphischen Serie; Numerierung mit 1                                                                                  |
| indicator_styleN     | <a href="#">int</a>    | Linienstil in der <a href="#">graphischen Serie</a> , der mittels des Werte aus <a href="#">ENUM_LINE_STYLE</a> angegeben wird. N - Nummer einer graphischen Serie, Numerierung faengt mit 1 an |
| indicator_typeN      | <a href="#">int</a>    | Typ der graphischen Darstellung, der durch den Wert aus <a href="#">ENUM_DRAW_TYPE</a> angegeben wird. N - Nummer einer graphischen Serie, Numerierung faengt mit 1 an                          |
| indicator_levelN     | <a href="#">double</a> | horizontaler Level N im selbsstaendigen Fenster des Indikators                                                                                                                                  |
| indicator_levelcolor | <a href="#">color</a>  | Farbe der horizontalen Levels des Indikators                                                                                                                                                    |
| indicator_levelwidth | <a href="#">int</a>    | Breite der horizontalen Levels des Indikators                                                                                                                                                   |
| indicator_levelstyle | <a href="#">int</a>    | Stil der horizontalen Niveaus des Indikators                                                                                                                                                    |
| script_show_confirm  |                        | Ausgabe des Bestaetigungsfensters vor dem Scriptablauf                                                                                                                                          |
| script_show_inputs   |                        | Ausgabe des Fensters mit Eigenschaften vor dem Scriptablauf und Verbot, Bestaetigungsfenster auszugeben                                                                                         |

|                  |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|------------------|-------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tester_indicator | <a href="#"><u>string</u></a> | Name des Benutzerindikators im Format " <i>Name_des Indicators.ex5</i> ". Die für Testenerforderliche Indikatoren werden automatisch aus Aufruf der Funktionen <a href="#"><u>iCustom()</u></a> bestimmt, wenn der entsprechende Parameter von der Konstantzeile vorgegeben ist. Für andere Fälle (Verwendung der Funktion <a href="#"><u>IndicatorCreate()</u></a> oder Verwendung einer nicht Konstantzeile im Parameter, der den Namen des Indikator vorgibt) ist diese Eigenschaft erforderlich. |
| tester_file      | <a href="#"><u>string</u></a> | Dateiname für Tester mit Angabe der Erweiterung, in Doppelanführungszeichen (als Konstantzeile). Die angegebene Datei wird dem Tester für Arbeit übertragen. Eingabeparameter für Testen müssen immer angegeben werden (wenn sie erforderlich sind)                                                                                                                                                                                                                                                  |
| tester_library   | <a href="#"><u>string</u></a> | Name der Bibliothek mit Erweiterung in Doppelanführungszeichen. Bibliothek kann sowohl mit der Erweiterung dll, als auch mit der Erweiterung ex5 sein kann. Die für Testen erforderliche Bibliotheken werden automatisch bestimmt. Aber wenn eine Bibliothek vom <a href="#"><u>Benutzerindikator</u></a> verwendet wird, muss man diese Eigenschaft verwenden.                                                                                                                                      |

#### Beispiel der Vorgabe der Beschreibung und Nummer der Version

```
#property version      "3.70"           // laufende Expertversion
#property description  "ZigZag universell mit Pesavento Patterns"
#property description  "Zum jetzigen Zeitpunkt werden in Indikator mehrere ZigZag mit v
#property description  "Es ist moeglich, eine grosse Anzahl anderer Indikatoren einzubau
```

```
#property description "Minima zeigen und aus diesen Minima und Maxima verschiedene gra
```



Beispiel der Andeutung einzelner Marke für jeden Indikatorpuffer ("C open;C high;C low;C close")

```
#property indicator_chart_window
#property indicator_buffers 4
#property indicator_plots 1
#property indicator_type1 DRAW_CANDLES
#property indicator_width1 3
#property indicator_label1 "C open;C high;C low;C close"
```



## Include Dateien (#include)

Befehlszeile *#include* kann in jeder Programmstelle auftreten, gewöhnlich aber befinden sich alle Einschliesse am Anfang der Datei des Ausgangstextes.

Aufrufformat:

```
#include <Dateiname>
#include "Dateiname"
```

### Beispiele:

```
#include <WinUser32.mqh>
#include "mylib.mqh"
```

Preprozessor ersetzt die Zeile *#include <Dateiname>* durch den Dateiinhalt WinUser32.mqh. Spitze Klammern bezeichnen, dass die Datei WinUser32.mqh dem Standardkatalog entnommen werden wird (gewöhnlich ist es Katalog\_des Terminals\MQL5\Include). Laufendes Katalog wird nicht durchgesehen.

Wenn der Dateiname in Anführungszeichen zteht, wird die Suche im laufenden Katalog (wo sich Hauptdatei des Ausgangstextes befindet) durchgeführt. Standardkatalog wird nicht durchgesehen.

### Sehen Sie auch

[Standardbibliothek](#), [Import der Funktionen](#)

## Import der Funktionen (#import)

Import der Funktionen wird von kompilierten Moduls MQL5 (Dateien \*.ex5) und aus Moduls des Operationssystems (Dateien \*.dll) durchgeführt. Modulname wird in der befehlenden Anweisung `#import` angegeben. Damit der Compiler den Aufruf der importierenden Funktion richtig durchführen und richtige [Parameterübergabe](#) gestalten kann, ist völlige Beschreibung der [Funktionen](#) notwendig. Funktionsbeschreibungen folgen unmittelbar der befehlenden Anweisung `#import "Modulname"`. Neuer Befehl `#import` (möglich auch ohne Parameter) beendet Beschreibungsblock der importierender Funktionen.

```
#import "Dateiname"
    func1 define;
    func2 define;
    ...
    funcN define;
#import
```

Zu importierende Funktionen können verschiedene Namen haben. Importiert kann aus verschiedenen Modulen der Funktion mit gleichen Namen gleichzeitig werden. Zu importierende Funktionen können Namen haben, die mit den Namen interner Funktionen zusammenfallen. Operation von [Kontexterlaubnis](#) bestimmt, welche Funktion aufgerufen muss.

Die Suchreihenfolge nach der Datei, die nach der Schlüsselwort `#import` angegeben ist, ist in der Sektion [Aufruf der Importfunktionen](#) beschrieben.

Da sich zu importierende Funktionen ausser compilierten Moduls befinden, kann der Compiler Richtigkeit der übertragenen Parameter nicht prüfen. Darum um Ausführungsfehler zu vermeiden muss man genau Inhalt und Ordnung der Parameter beschreiben, die in importierte Funktionen übertragen werden. Parameter, übertragen in importierte Funktionen (sowie aus EX5, als auch aus DLL-Modulen) können Default-Werte haben.

In importierten Funktionen können als Parameter nicht verwendet werden:

- [Anzeiger](#) (\*);
- Verweise auf Objekte, die [dynamische Felder](#) und/oder Anzeiger enthalten.

In die aus DLL importierten Funktionen können als Parameter keine Klassen, Zeilenfelder oder komplizierte Objekte übertragen werden, die Zeilen und/oder dynamische Felder verschiedener Typen enthalten.

**Beispiele:**

```
#import "user32.dll"
int    MessageBoxW(uint hWnd,string lpText,string lpCaption,uint uType);
#import "stdlib.ex5"
string ErrorDescription(int error_code);
int    RGB(int red_value,int green_value,int blue_value);
bool   CompareDoubles(double number1,double number2);
string DoubleToStrMorePrecision(double number,int precision);
string IntegerToHexString(int integer_number);
#import "ExpertSample.dll"
int    GetIntValue(int);
double GetDoubleValue(double);
```

```
string GetStringValue(string);  
double GetArrayItemValue(double &arr[], int, int);  
bool SetArrayItemValue(double &arr[], int, int, double);  
double GetRatesItemValue(double &rates[][6], int, int, int);  
#import
```

für Importieren der Funktionen während der Durchführung eines mql5-Programms wird frühe Bindung verwendet. Das bedeutet, dass Bibliothek während der Ladung des ex5-Programm geladen wird, das sie verwendet.

Es ist nicht empfehlenswert, den vollständig qualifizierten Namen des geladenen Moduls des Typs *Drive:\Directory\FileName.Ext* zu verwenden. Bibliotheken MQL5 werden vom Ordner *terminal\_dir\MQL5\Libraries* geladen.

Sehen Sie auch

[Include Dateien](#)



## Bedingte Kompilierung (#ifdef, #ifndef, #else, #endif)

Durch Direktiven für die bedingte Kompilierung von Präprozessor können Sie Teil des Programms kompilieren oder vermissen, abhängig von der Erfüllung bestimmter Bedingungen.

Der Bedingung kann eine der nachfolgend beschriebenen Formen annehmen.

```
#ifdef identifier
    // Code hier wird kompiliert, wenn identifier bereits für die Präprozessor in Befehl #define
#endif

#ifndef identifier
    // Code hier wird kompiliert, wenn identifier für die Präprozessor in Befehl #define
#endif
```

Jedem der Befehlen der bedingten Kompilierung kann eine beliebige Anzahl von Zeilen folgen, die möglicherweise der Befehl #else mit Ende #endif enthalten. Wenn die Bedingung erfüllt ist, werden die Zeilen zwischen #else und #endif ignoriert. Ist die Bedingung nicht erfüllt, dann alle Zeilen zwischen dem Prüfung und Befehl #else ignoriert werden. Wenn es dieser Befehl gibt nicht, dann zwischen dem Prüfung und dem Befehl #endif.

### Beispiel:

```
#ifndef TestMode
    #define TestMode
#endif

//+-----+
//| Script program start function |
//+-----+

void OnStart()
{
    #ifdef TestMode
        Print("Test mode");
    #else
        Print("Normal mode");
    #endif
; }
```

Abhängig vom Typ des Programms und Standardkompilierungsmodus, werden Makros wie folgt definiert:

Makro `__MQL5__` ist bei der Kompilierung der Datei `*.mq5` verfügbar. Bei der Kompilierung von `*.mq4` ist Makro `__MQL4__` verfügbar.

Makro `_DEBUG` ist bei der Kompilierung für die Fehlersuche verfügbar.

Makro `_Release` ist bei der Kompilierung für anders als Fehlersuche verfügbar.

### Beispiel:

```
//+-----+
//| Script program start function |
//+-----+

void OnStart()
```

```
{
#ifdef __MQL5__
#ifdef _DEBUG
    Print("Hello from MQL5 compiler [DEBUG]");
#else
#ifdef _RELEASE
    Print("Hello from MQL5 compiler [RELEASE]");
#endif
#endif
#else
#ifdef __MQL4__
#ifdef _DEBUG
    Print("Hello from MQL4 compiler [DEBUG]");
#else
#ifdef _RELEASE
    Print("Hello from MQL4 compiler [RELEASE]");
#endif
#endif
#endif
#endif
#endif
}
```

## Objektorientiertes Programmieren

Objektorientiertes Programmieren ist datenorientiertes Programmieren, wobei Daten und Verhalten unaufloslich miteinander verbunden sind. Zusammen sind Daten und Verhalten eine Klasse, und Objekte sind Klassenexemplare.

Bestandteile des objektorientierten Programmieren sind:

- [Inkapsulation und Erweiterungsfähigkeit der Typen](#)
- [Vererbung](#)
- [Polimorphismus](#)
- [überladung](#)
- [Virtuelle Funktionen](#)

Objektorientiertes Programmieren betrachtet Berechnungen als Verhaltensmodellierung. Das was modelliert wird sind Objekte, dargestellt durch rechnerische Abstraktionen. ZB. wollen wir das allgemeinbekannte Spiel "Tetris" schreiben, dafür muessen wir lernen, die Erscheinung einer Zufallfigur zu modellieren, die aus vier Kaestchen besteht, miteinander durch Ränder verbunden. Auch muss man die Fallgeschwindigkeit der Figur regeln, Rotierungs- und Schubsoperationen der Figur einstellen. Umsetzungen der Figur auf Display sind durch Glasgrenzen begrenzt, diese Forderung muessen wir auch modellieren. Außerdem muessen die gefuellten Reihen der Kaestchen im Glas entfernt werden und die erarbeiteten Punkte berechnet werden muessen.

So, fordert dieses einfaches Spiel das Schaffen vo mehreren Modellen - Figurmodell, Glasmodell, Modell der Figurbewegung im Glas usw. Alle diesen Modelle sind Abstraktionen, dargestellt von Berechnungen im Computer. Für Beschreibung dieser Modelle wird der Begriff abstrakter Datentyp ADT (oder [komplexer Datentyp](#)) verwendet. Streng gesagt ist Bewegungsmodell der "Figur" im "Glas" kein Datentyp, sondern Gesamtheit von Datenoperationen des Typs "Figur", die Datenbegrenzungen des Typs "Glas" verwenden.

Objekte sind Variablen der [Klasse](#). Objektorientiertes Programmieren ermöglicht abstrakte Datentypen einfach zu erzeugen und zu verwenden. Objektorientiertes Programmieren verwendet Mechanismus von [Vererbung](#). Vererbung ist deswegen vorteilhaft, weil es die Erhaltung von sekundaeren Typen aus den vom Benutzer bestimmten Datentypen ermöglicht. für die Figurerzeugung im Tetris ist es bequem, vor allem Basisklasse Shape zu erzeugen, auf deren Grund sekundaere Typen aller sieben moeglichen Figuren in Tetris erhalten werden. In der Basisklasse ist Figurverhalten bestimmt, in den sekundaeren ist Realisierung des Verhaltens jeder konkreten Figur detailliert.

In OPP sind Objekte für Ihr Verhalten verantwortlich. Entwickler von ADT muss darin Code für die Beschreibung jedes Verhaltens einbeschliessen, das von den entsprechenden Objekten zu erwarten ist. Die Tatsache, dass das Objekt für sein Verhalten selbst verantwortlich ist, vereinfacht bedeutend das Programmieren für Benutzer dieses Objektes.

Wenn wir auf dem Display Figur zeichnen wollen, muessen wir kennen, wo sich ihr Zentrum befunden wird und wie man sie zeichnen muss. Wenn es einzelne Figur genau kennt, wie sie sich zeichnen muss, muss der Programmierer bei der Verwendung dieser Figur dem Objekt die Nachricht "sich zeichnen" übertragen.

Sprache MQL5 ist der Sprache C++ aehnlich und darin ist auch der Mechanismus der [Inkapsulation](#) für Realisierung von ADT realisiert. Inkapsulation vereinigt in sich einerseits interne Details der Realisierung des konkreten Typs und andererseits die aussen zugaenglichen Funktionen, die die

Objekte des Typs bewirken können. Realisierungsdetails können für das Programm, das diesen Typ verwendet, unzugänglich sein.

Zum Begriff OPP gehört eine Reihe von Konzeptionen einschliesslich folgende:

- Modellieren der Handlungen aus der realen Welt
- Vorhandensein von Datentypen bestimmt vom Benutzer
- Verhüllung der Realisierungsdetails
- Möglichkeit von mehrmaliger Verwendung durch Vererbung
- Interpretieren der Funktionsaufrufe in der Durchführungsetappe

Einige von diesen Begriffen sind ganz undeutlich, einige - abstrakt, andere haben allgemeinen Charakter.

## Inkapsulation und Erweiterungsfaehigkeit der Typen

OOP ist balancierter Zugang zum Schreiben von Software. Daten und Verhalten sind zusammen verpackt. Diese Inkapsulation erzeugt die vom Benutzer bestimmten Datentypen, die eigene Sprachdatentypen und mit ihnen zusammenwirkende Typen erweitern. Erweiterungsfaehigkeit der Typen ist die Moeglichkeit, die vom Benutzer bestimmten Datentypen der Sprache zuzusetzen, die so einfach verwendet werden können, wie [Grundtypen](#).

Abstrakter Datentyp, zB. die Zeile ist die Beschreibung des idealen, allgemeinbekannten Verhaltens des Typs. Benutzer der Zeile weiss, dass Operationen, wie Konkatenanz oder Drucken bestimmtes Verhalten hat. Operationen on Konkatenanz und Drucken heissen Methoden.

Bestimmte ADT Realisierung kann Beschraenkungen haben; die Laenge von Zeilen kann ZB. begrenzt werden. Diese Beschraenkungen beeinflussen Verhalten, offen für alle. Zu gleicher Zeit beeinflussen interne oder geschlossene Realisierungsdetails das nicht, wie der Benutzer das Objekt sieht. ZB. die Zeile wird oft als Feld realisiert; dabei ist interne Grundadresse der Elemente dieses Feldes und sein Name für Benutzer nicht bedeutend sind.

Inkapsulation ist die Faehigkeit, interne Details zu verhuellen Versorgung mit dem offenen Interface zu dem vom Benutzer bestimmten Typ. In MQL5 werden wie in C++, Definitionen der Klasse und der Struktur ([class](#) und [struct](#)) mit den Schluesselwoertern des Zuganges [private](#) (geschlossen), [protected](#) (geschuetzt) und [public](#) (offen) verwendet.

Schluesselwort [public](#) zeigt, dass Zugang zu Gliedern, die ihm folgen, ist offen ohne Begrenzungen. Ohne dieses Schluesselwort sind Klassenglieder als Standardeinstellung geschlossen. Geschlossene Glieder sind erst den Funktionen-Gliedern der eigenen Klasse.

Geschuetzte Klassenglieder sind nicht nur für die Funktionen-Glieder der eigener Klasse, sondern auch für die Klassen-Nachfolger zugaenglich. Offene Glieder sind für jede Funktion innerhalb des Sichtbereiches der Klassenerklärung zugaenglich. Geschlossenheit ermöglicht Teil der Klassenrealisierung zu verhuellen, und verhindert damit nicht vorausgesehene Änderungen der Datenstruktur. Zugangsbegreuzung oder Datenverhuellung ist eine Eigenschaft des objektorientiertes Programmieren.

Gewoehnlich versucht man, Klassenglieder zu schuetzen und sie mit dem Modifikator [protected](#) zu erklären, Einstellung und Lesen der Werte dieser Glieder ist durch sogenannte [set-](#) und [get-](#)Methoden durchgeführt, die vom Zugangsmodifikator , [public](#) bestimmt werden

### Beispiel:

```
class CPerson
{
protected:
    string          m_name;                // Name
public:
    void            SetName (string n) {m_name=n;} // stellt Name ein
    string          GetName () {return (m_name);} // gibt Name zurück
};
```

Solcher Zugang hat eine Reihe der Vorteile. Erstens kann nach dem Namen der Funktion verstanden werden, was sie macht - ob sie den Wert des Klassenglieders einstellt oder erhaelt. Zweitens können wir in der Zukunft Typ der Variable `m_name` in der Klasse `CPerson` oder in einer seiner sekundaeren

Klasse verändert werden, wenn erforderlich.

In diesem Fall ist es genug, Realisierung der Funktionen SetName() und GetName() zu verändern, die Objekte der Klasse CPerson können im Programm ohne Änderungen im Code verwendet werden, denn der Benutzer wird nicht wissen, dass sich Datentyp m\_name verändert hat.

**Beispiel:**

```
struct Name
{
    string      first_name;           // Vorname
    string      last_name;           // Name
};

class CPerson
{
protected:
    Name        m_name;              // Vorname
public:
    void        SetName(string n);
    string      GetName() {return(m_name.first_name+" "+m_name.last_name);}
private:
    string      GetFirstName(string full_name);
    string      GetLastName(string full_name);
};

void CPerson::SetName(string n)
{
    m_name.first_name=GetFirstName(n);
    m_name.last_name=GetLastName(n);
};

string CPerson::GetFirstName(string full_name)
{
    int pos=StringFind(full_name, " ");
    if(pos>0) StringSetCharacter(full_name,pos,0);
    return(full_name);
};

string CPerson::GetLastName(string full_name)
{
    string ret_string;
    int pos=StringFind(full_name, " ");
    if(pos>0) ret_string=StringSubstr(full_name,pos+1);
    else     ret_string=full_name;
    return(ret_string);
};
```

**Sehen Sie auch**

Datentypen

## Vererbung

Eigenartigkeit von OOP besteht in der Aufmunterung zu Wiederverwendung des Codes mittels Vererbung. Die neue Klasse wird von der vorausgehenden erzeugt, die Basisklasse genannt wird. Sekundäre Klasse verwendet die Glieder der Basisklasse, kann aber sie verändern und erweitern.

Viele Typen sind Variationen der vorhandenen Typen. Oft ist es ermüdend einen neuen Kodex für jeden auszuarbeiten. Außerdem erscheinen mit neuem Code neue Fehler. Sekundäre Klasse erbt die Beschreibung der Basisklasse und macht es unnötig, Code wieder auszuarbeiten und zu testen. Vererbungsverhalten sind hierarchisch.

Hierarchie ist die Methode, die Elemente in in voller Vielfältigkeit und Kompliziertheit zu kopieren ermöglicht. Sie leitet Objektklassifizierung ein. ZB gibt es im periodischen System von Elementen Gase. Sie haben Eigenschaften, die allen Systemelementen eigen sind.

Neutrale Gase stellen die nächste wichtige Subklasse dar. Hierarchie besteht darin, dass neutrales Gas, zB. Argon ist ein Gas, und Gas ist ein Systemelement. Solche Hierarchie ermöglicht das Verhalten von neutralen Gasen leicht zu interpretieren. Wir wissen, dass ihre Atome Protonen und Elektronen haben, was für andere Elemente auch typisch ist.

Wir wissen, dass sie bei Raumtemperatur gasförmig sind, wie alle Gase. Wir wissen, dass kein Gas aus der Subklasse neutrale Gase mit anderen Elementen reagiert und das ist die Eigenschaft aller neutralen Gase.

Betrachten wir Vererbung am Beispiel mit geometrischen Figuren. Für Beschreibung aller Vielfältigkeit von einfachen Figuren (Kreis, Dreieck, Rechteck, Quadrat usw.) ist es am besten eine Basisklasse zu erzeugen ([ADT](#)), die Vorfahr aller sekundären Klassen ist.

Erzeugen wir eine Basisklasse CShape, in der es nur allgemeine Glieder gibt, die die Figur beschreiben. Diese Glieder beschreiben die Eigenschaften, die jeder Figur eigen sind - Figurtyp und Koordinaten des Bezugspunktes.

### Beispiel:

```
//--- Basisklasse Figur
class CShape
{
protected:
    int      m_type;           // Figurtyp
    int      m_xpos;           // X - Koordinate des Bezugspunktes
    int      m_ypos;           // Y - Koordinate des Bezugspunktes
public:
    CShape() {m_type=0; m_xpos=0; m_ypos=0;} // Konstruktor
    void     SetXPos(int x) {m_xpos=x;} // stellen wir X ein
    void     SetYPos(int y) {m_ypos=y;} // stellen wir Y ein
};
```

Weiter erzeugen wir von der Basisklasse sekundäre Klassen, in die wir notwendige Felder zufügen, die jede konkrete Klasse präzisieren. Für die Figur Circle(Kreis) ist es notwendig das Glied zuzufügen, das Radiuswert enthält. Figur Quadrate (Quadrat) ist durch den Wert der Quadratseite charakterisiert. Darum werden sekundäre Klassen, die von der Basisklasse CShape geerbt werden, folgenderweise erklärt :



```
//--- sekundaere Klasse Kreis
class CCircle : public CShape      // nach Doppelpunkt wird Basisklasse angegeben,
{                                  // von der Vererbung erfolgt
private:
    int          m_radius;        // Kreisradius

public:
    CCircle(){m_type=1;} // Konstruktor, Typ ist 1 gleich
};
```

für Quadrat sieht Klassenerklärung ähnlicherweise aus:

```
//--- sekundaere Klasse Quadrat
class CSquare : public CShape      // nach Doppelpunkt wird Basisklasse angegeben ,
{                                  // von der Vererbung erfolgt
private:
    int          m_square_side;    // Quadratseite

public:
    CSquare(){m_type=2;} // Konstruktor ist 2 gleich
};
```

Es muss bemerkt werden, dass bei der Erzeugung eines Objektes vor allem Konstruktor einer Basisklasse, und dann [Konstruktor](#) einer sekundären Klasse aufgerufen wird. Bei der Entfernung eines Objektes wird zuerst [Destruktor](#) einer sekundären Klasse und dann Destruktor einer Basisklasse aufgerufen.

So nach der Erklärung allgemeiner Glieder in einer Basisklasse können wir in sekundäre Klassen zusätzliche Glieder hinzufügen, die eine konkrete Klasse präzisieren. Vererbung erlaubt große Kodebibliotheken zu erzeugen, die mehrmals und wiederholt verwendet werden können.

Syntax der Erzeugung der sekundären Klasse von der schon vorhandenen sieht so aus:

```
class Klassenname :
    (public | protected | private) opt Name der Basisklasse
{
    Gliedererklärungen
};
```

Einer der Aspekte der sekundären Klasse ist Sichtbarkeit (Offenheit) seiner Glieder-Nachfolger. Schlüsselwörter public, protected und private werden verwendet, um anzudeuten, inwieweit Glieder der Basisklasse für Glieder der sekundären Klasse zugänglich sind. Verwendung im Namen der sekundären Klasse der Schlüsselklasse public, der nach Doppelpunkt folgt, bedeutet, dass geschützte und offene (protected und public) Glieder der Basisklasse CShape wie geschützte und offene Glieder der sekundären Klasse CCircle geerbt werden müssen.

Geschlossene Glieder der Basisklasse sind für sekundäre Klasse nicht zugänglich. Offene Vererbung bedeutet auch, dass sekundäre Klassen (CCircle und CSquare) CShape sind. D.h. Quadrat (CSquare) ist eine Figur (CShape), eine Figur muss aber nicht unbedingt ein Quadrat sein.

Sekundäre Klasse ist Modifikation der Basisklasse; sie erbt geschützte und offene Glieder der Basisklasse. Erst Konstruktors und Destruktors der Basisklasse können nicht geerbt werden. Oft

werden in die sekundäre Klasse neue Glieder zu Gliedern der Basisklasse hinzugefügt.

Sekundäre Klasse kann Realisierung der Funktionen-Glieder einschließen, die sich von der Basisklasse unterscheidet. Das hat nichts mit Überladung zu tun, wenn der Sinn desselben Funktionsnamens verschieden für verschiedene Signaturen sein kann.

Beim geschützten Vererbung werden offene und geschützte Glieder der Basisklasse zu offenen und geschützten Gliedern der sekundären Klasse. Beim geschlossenen Vererbung werden offene und geschützte Glieder der Basisklasse zu geschlossenen Gliedern der sekundären Klasse.

Beim geschützten und geschlossenen Vererbung stimmt es nicht, dass Objekt der sekundären Klasse Objekt der Basisklasse ist. Geschütztes und geschlossenes Vererbung treten selten auf und müssen sehr vorsichtig verwendet werden.

Es sollte klar sein, dass die Art der Vererbung (public, protected oder private) beeinflusst nicht die Mittel für den Zugang an die Glieder der Basisklassen in der Vererbungshierarchie der abgeleiteten Klasse (Erben). In jeder Art der Vererbung, nur Glieder einer Basisklasse, die mit dem Zugang-Spezifizierer public und protected deklariert worden, sind aus abgeleiteten Klassen erreichbar. Betrachten wir das obige Beispiel:

```
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

//+-----+
//| Beispielklasse mit mehreren Arten des Zugriffs |
//+-----+

class CBaseClass
{
private:          //--- Privater Mitglied ist von den abgeleiteten Klassen nicht zu
    int           m_member;
protected:      //--- Geschützter Mitglied ist von der Basisklasse und ihren abge
    int           Member() {return(m_member); }
public:          // Konstruktor der Klasse ist allen zugänglich
    CBaseClass() {m_member=5;return;};
private:        // Private Methode für die Zuweisung eines Wertes an den Mitglied
    void          Member(int value) { m_member=value;};

};

//+-----+
//| Die abgeleitete Klasse mit Fehlern |
//+-----+

class CDerivaed: public CBaseClass // Es ist nicht notwendig public-Vererbung anzugeben
{
public:
    void Func() // In der abgeleiteten Klasse definieren wir eine Funktion mit Aufrufer
    {
        //--- Versuch private Mitglied der Basisklasse zu ändern
        m_member=0;          // Fehler, ist ein privates Mitglied der Basisklasse nicht zu
        Member(0);          // Fehler, ist ein privates Mitglied der Basisklasse in der
        //--- Lesen einer Basisklassenmitglied
```

```

    Print(m_member); // Fehler, ist ein privates Mitglied der Basisklasse nicht zu
    Print(Member()); // Kein Fehler, offenes Mitglied der Basisklasse ist immer zu
};
};

```

In diesem Beispiel hat Klasse CBaseClass nur eine öffentliche Methode - den Konstruktor. Konstruktoren sind automatisch aufgerufen, wenn ein Objekt der Klasse erstellt wird. So kann man nicht von außen den privaten Mitglieder m\_member und die geschützte Methode Member() aufrufen. Aber mit einer offenen (public) Vererbung ist die Methode der Basisklasse Member() von den abgeleiteten Klassen zugänglich.

In der geschützte (protected) Vererbung, werden alle Basisklassenmitglieder mit einem offenen und geschützten Zugriff geschützt. Dies bedeutet, dass, wenn das öffentliche Datenelemente und Methoden der Basisklasse von außen zugänglich sind, dann in geschützter Vererbung sind sie nur aus den Klassen des Kindes und seiner anschließenden abgeleiteten Klassen zugänglich.

```

//+-----+
//| Beispielklasse mit mehreren Arten des Zugriffs |
//+-----+
class CBaseMathClass
{
private: //--- Privater Mitglied ist von den abgeleiteten Klassen nicht zu
    double m_Pi;
public: //--- Erhalten und angeben Wert für m_Pi
    void SetPI(double v){m_Pi=v;return;};
    double GetPI(){return m_Pi;};
public: // Konstruktor der Klasse ist allen zugänglich
    CBaseMathClass(){SetPI(3.14); PrintFormat("%s",__FUNCTION__);};
};

//+-----+
//| Abgeleitete Klasse, in deren m_Pi kann nicht verändert werden |
//+-----+
class CProtectedChildClass: protected CBaseMathClass // Geschützte Vererbung
{
private:
    double m_radius;
public: //--- Offene Methoden in der abgeleiteten Klasse
    void SetRadius(double r){m_radius=r; return;};
    double GetCircleLength(){return GetPI()*m_radius;};
};

//+-----+
//| Funktion started den Skript |
//+-----+
void OnStart()
{
//--- Bei der Erstellung des Kindes wird Basisklassenkonstruktor automatisch aufgerufen
    CProtectedChildClass pt;
//--- Radius angeben
    pt.SetRadius(10);
    PrintFormat("Length=%G",pt.GetCircleLength());
}

```

```
//--- Wenn den folgenden String benutzen, bekommen wir eine Fehlermeldung beim Kompilieren
// pt.SetPI(3);

//--- und jetzt deklarieren wir eine Variable einer Basisklasse und versuchen, eine Konstruktor zu rufen
CBaseMathClass bc;
bc.SetPI(10);
//--- Ergebnis
PrintFormat("bc.GetPI()=%G",bc.GetPI());
; }
```

Dieses Beispiel zeigt, dass die Methoden SetPI() und GetPI() in der Basisklasse CBaseMathClass offen sind und von überall angerufen werden können. Aber zur gleichen Zeit für sein Kind CProtectedChildClass können die Aufrufe dieser Methoden nur aus Methoden der Klasse CProtectedChildClass oder ihren Kinder gemacht werden.

Mit einer privaten Vererbung werden alle Mitglieder der Basisklasse mit Zugriff public und protected geschlossen, und mit weiterer Vererbung ist Aufruf dieser Mitglieder unmöglich.

In MQL5 gibt es keine multiple Vererbung.

Sehen Sie auch

[Strukturen und Klassen](#)

## Polymorphismus

Polymorphismus ist die Möglichkeit für Objekte verschiedener Klassen verbunden durch Vererbung beim Zugriff zu derselben Funktion-Element verschieden zu reagieren. Das hilft, universelle Mechanismen zu schaffen, die nicht nur das Verhalten der Basisklasse beschreiben, sondern auch das Verhalten der Klassen-Nachfolger.

Setzen wir die Ausarbeitung der Basisklasse CShape fort, in der wir die Funktion-Glied GetArea() definieren, die für Berechnung der Figurflaeche bestimmt ist. In allen Klassen-Nachfolger, erzeugt durch Vererbung von der Basisklasse, definieren wir die Funktion entsprechend den Regeln der Flaechenberechnung für konkrete Figur um.

für Quadrat (Klasse CSquare) wird die Flaeche durch Seiten, für Kreis (Klasse CCircle) durch Radius berechnet usw. Wir können Feld schaffen für Speichern von Objeketen des Typs CShape, in dem sowohl das Objekt der Basisklasse, als auch alle ihre Nachfolger gespeichert werden können. Im Nachhinein können wir dieselbe Funktion für jedes Element dieses Feldes aufrufen.

### Beispiel:

```
//--- Basisklasse
class CShape
{
protected:
    int         m_type;           // Figurtyp
    int         m_xpos;           // X - Koordinate des Bezugspunktes
    int         m_ypos;           // Y - Koordinate des Bezugspunktes
public:
    void        CShape() {m_type=0;}; // Konstruktor, Typ ist Null gleich
    int         GetType() {return(m_type);}; // gibt Figurtyp zurück
    virtual     double    GetArea() {return (0); } // gibt Figurflaeche zurück
};
```

Jetzt haben alle Variablen die Funktion-Glied getArea(), die Nullwert zurückgibt. Realisierung dieser Funktion wird in jedem Nachfolger verschieden sein.

```
//--- sekundaere Klasse Kreis
class CCircle : public CShape // nach Doppelpunkt wird Basisklasse angegeben
{                               // von der Vererbung erfolgt
private:
    int         m_radius;        // Kreisradius
public:
    void        CCircle() {m_type=1;}; // Konstruktor, Typ ist 1 gleich
    void        SetRadius(double r) {m_radius=r;};
    virtual     double    GetArea() {return (3.14*m_radius*m_radius);} // Kreisflaeche
};
```

Für Quadrat sieht Klassenerklärung ähnlich aus:

```
//--- sekundaere Klasse Quadrat
class CSquare : public CShape // nach Doppelpunkt wird Basisklasse angegeben
```

```

{
    // von der Vererbung erfolgt
private:
    int        m_square_side;    // Quadratseite

public:
    void        CSquare(){m_type=2;}; // Konstruktor, Typ ist 2 gleich
    void        SetSide(double s){m_square_side=s;};
    virtual double GetArea(){return (m_square_side*m_square_side);} //Quadratinhalt
};

```

Da für Berechnung des Quadrats und des Kreises entsprechende Werte der Glieder `m_radius` und `m_square_side` notwendig sind, haben wir in der Erklärung der entsprechenden Klassen Funktionen `SetRadius` und `SetSide()` zugesetzt.

Es wird davon ausgegangen, dass in unserem Programm Objekte der verschiedenen Typen (`CCircle` und `CSquare`) abgeleitet von einem Basistyp `CShape` verwendet werden. Polymorphismus erlaubt die Erstellung eines Arrays von Objekten der Basisklasse `CShape`, aber bei der Deklaration von diesem Array sind diese Objekte noch nicht bekannt und ihre Typ nicht definiert ist.

Die Entscheidung, welcher Typ von Objekt wird in jedem Element des Arrays enthalten sein werden direkt während der Programmausführung genommen werden. Dies beinhaltet die [dynamische Erstellung](#) von Objekten der entsprechenden Klassen, und damit die Notwendigkeit, [Objekt Zeiger](#) anstelle von Objekten zu verwenden..

Das [new](#)-Array ist für die dynamische Erzeugung von Objekten eingesetzt. Jedes solches Objekt muss einzeln und explizit mit dem Operator [delete](#) gelöscht werden. Deshalb erklären wir ein Array den Zeigern von `CShape` Typ, und erstellen wir ein Objekt einer richtigen Typ für jedes Element (**new Klassename**), wie im folgenden Skript Beispiel gezeigt:

```

//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    //--- Erklären wir Array von Zeigern auf Objekte der Basistyp
    CShape *shapes[5];    // Array von Zeigern auf Objekte CShape

    //--- Hier füllen wir das Array mit abgeleiteten Objekten
    //--- Erklären wir Zeigern auf Objekt der CCircle-Typ
    CCircle *circle=new CCircle();
    //--- Stellen wir die Eigenschaften von Objekt im circle Zeiger auf
    circle.SetRadius(2.5);
    //--- Platzieren wir den Zeigerwert in shapes[0]
    shapes[0]=circle;

    //--- Erstellen wir noch ein Objekt CCircle und schreiben sein Zeiger in shapes[1]
    circle=new CCircle();
    shapes[1]=circle;
    circle.SetRadius(5);

    //--- Hier haben wir absichtlich "vergessen ", einen Wert für shapes[2] zu geben

```

```

//circle=new CCircle();
//circle.SetRadius(10);
//shapes[2]=circle;

//--- Setzen wir NULL für die nicht genutzte Element
    shapes[2]=NULL;

//--- Erstellen wir Objekt CSquare und schreiben sein Zeiger in shapes[3]
    CSquare *square=new CSquare();
    square.SetSide(5);
    shapes[3]=square;

//--- Erstellen wir CSquare und schreiben sein Zeiger in shapes[4]
    square=new CSquare();
    square.SetSide(10);
    shapes[4]=square;

//--- Es gibt ein Array von Zeiger, erhalten wir sein Größe
    int total=ArraySize(shapes);
//--- Passen wir in der Schleife über alle Zeiger in dem Array
    for(int i=0; i<5;i++)
    {
        //--- Wenn der Zeiger an dem angegebenen Index gültig ist
        if(CheckPointer(shapes[i])!=POINTER_INVALID)
        {
            //--- Speichern wir Typ und Fläche der Figur in Logs
            PrintFormat("Objekt von Typ %d mit Fläche %G",
                shapes[i].GetType(),
                shapes[i].GetArea());

            ;}
        //--- Wenn der Zeiger ist vom Typ POINTER_INVALID
        else
        {
            //--- Melden wir einen Fehler
            PrintFormat("Objekt shapes[%d] ist nicht initialisiert! Sein Zeiger ist %s",
                i,EnumToString(CheckPointer(shapes[i])));

            ;}
        ;}

//--- Wir müssen zerstören alle erstellten dynamischen Objekten
    for(int i=0;i<total;i++)
    {
        //--- Wir können nur die Objekte die den Zeiger vom typ POINTER_DYNAMIC haben löschen
        if(CheckPointer(shapes[i])==POINTER_DYNAMIC)
        {
            //--- Melden wir die Löschung
            PrintFormat("Löschen wir shapes[%d]",i);
            //--- Löschen wir ein Objekt durch seinen Zeiger
            delete shapes[i];

```

```
    ; }  
    ; }  
    ; }
```

Beachten Sie, dass wenn das Object mit dem Operator [delete](#) gelöscht wird, [der Typ seinen Zeiger](#) überprüfen werden soll. Nur Objekte mit dem Zeiger [POINTER\\_DYNAMIC](#) kann durch delete gelöscht werden. Für andere Typen wird ein Fehler zurückgegeben.

Ausser Umdefinieren der Funktion bei der Vererbung schließt Polymorphismus auch Realisierung derselben Funktion mit verschiedenem Parametervorrat innerhalb einer Klasse. Das bedeutet in der Klasse können mehrere Funktionen mit demselben Namen, aber verschiedenem Parametertyp/Parametervorrat definiert werden. In diesem Fall wird Polymorphismus durch [Funktionüberladen](#) realisiert.

Sehen Sie auch

[Standardbibliothek](#)



## Überladen

Innerhalb einer Klasse können zwei oder mehrere Methoden festgestellt werden, die zusammen denselben Namen verwenden, aber verschiedene Parameterzahl haben. Wenn es der Fall ist, heißen die Methoden *überladene* Methoden und der Prozess heißt *überladen der Methode*. Überladen der Methode ist eines der Verfahren, mit dessen Hilfe [Polymorphismus](#) realisiert wird. Überladen der Methoden in Klassen erfolgt nach denselben Regeln, wie [überladen der Funktionen](#).

Wenn es für zu aufgerufene Funktion kein genaues Passen gibt, sucht der Compiler konsequent die passende Funktion nach drei Niveaus:

1. Suche unter Klassenmethoden;
2. Suche unter Methoden der Basisklassen, konsequent vom nächsten Vorfahren bis zum allerersten;
3. Suche unter anderen Funktionen.

Wenn es auf keinem Niveau genaues Passen gefunden ist, aber einige passende Funktionen auf verschiedenen Niveaus gefunden werden, wird die Funktion verwendet, die auf dem kleinsten Niveau gefunden ist. Innerhalb eines Niveaus kann nicht mehr als eine passende Funktion sein.

In MQL5 gibt es kein überladen von Anweisungen.

Sehen Sie auch

[Überladung der Funktionen](#)

## Virtuelle Funktionen

Das Schlüsselwort `virtual` dient als Funktionsspezifikator, der Mechanismus für dynamische Auswahl in der Etappe der Ausführung der passenden Funktion-Glied unter Funktionen der Basis- und sekundaeren Klasse, Strukturen können keine virtuelle Funktionen haben. Es kann erst für die Änderung [Erklärung](#) der Funktionen-Glieder verwendet werden.

Virtuelle Funktion muss wie eine normale Funktion [ausführenden\\_Koerper](#) haben. Beim Aufruf hat sie dieselbe Semantik, wie andere Funktionen. .

Virtuelle Funktion kann in der sekundaeren Klasse substituiert werden. Die Auswahl der [Funktionsdefinition](#) erfolgt dynamisch (in der Ausführungsetappe). Typischer Fall - wenn die Basisklasse virtuelle Funktion hat, und sekundaere Klassen ihre eigenen Versionen dieser Funktion haben.

Anzeiger auf die Basisklasse kann entweder auf das Objekt der Basisklasse oder auf das Objekt der sekundaeren Klasse hinweisen. Auswahl der aufgerufenen Funktion-Glied wird in der Ausführungsetappe durchgeführt und wird vom Objekttyp und nicht vom Anzeigertyp abhängig. Beim Fehlen des Gliedes des sekundaeren Typs wird vorbestimmt virtuelle Funktion der Basisklasse verwendet.

[Destruktors](#) sind immer virtuell unabhängig davon, ob sie mit dem Schlüsselwort `virtual` erklärt werden oder nicht.

Betrachten wir die Verwendung von virtuellen Funktionen am Beispiel des Programms MT5\_Tetris.mq5. In einschliessender Datei MT5\_TetisShape.mqh ist die Basisklasse CTetrisShape mit der virtuellen Funktion Draw (zeichnen) bestimmt.

```
//+-----+
class CTetrisShape
{
protected:
    int          m_type;
    int          m_xpos;
    int          m_ypos;
    int          m_xsize;
    int          m_ysize;
    int          m_prev_turn;
    int          m_turn;
    int          m_right_border;
public:
    void          CTetrisShape();
    void          SetRightBorder(int border) { m_right_border=border; }
    void          SetYPos(int ypos)         { m_ypos=ypos;           }
    void          SetXPos(int xpos)         { m_xpos=xpos;           }
    int          GetYPos()                  { return(m_ypos);         }
    int          GetXPos()                  { return(m_xpos);         }
    int          GetYSize()                 { return(m_ysize);        }
    int          GetXSize()                 { return(m_xsize);        }
    int          GetType()                  { return(m_type);         }
    void          Left()                    { m_xpos-=SHAPE_SIZE;     }
```

```

void      Right()      { m_xpos+=SHAPE_SIZE;      }
void      Rotate()     { m_prev_turn=m_turn; if(++m_turn>3) m_turn=0; }
virtual void Draw()     { return;                }
virtual bool CheckDown(int& pad_array[]);
virtual bool CheckLeft(int& side_row[]);
virtual bool CheckRight(int& side_row[]);
};

```

Weiters wird diese Funktion für jede sekundäre Klasse realisiert entsprechend den Besonderheiten der Klasse-Nachfolger. ZB. hat die erste Figur CTetrisShape1 ihre eigene Realisierung der Funktion Draw():

```

class CTetrisShape1 : public CTetrisShape
{
public:
    ///--- Zeichnung der Figur
    virtual void Draw()
    {
        int i;
        string name;
        ///---
        if(m_turn==0 || m_turn==2)
        {
            ///--- horizontaler Stock
            for(i=0; i<4; i++)
            {
                name=SHAPE_NAME+(string)i;
                ObjectSetInteger(0,name,OBJPROP_XDISTANCE,m_xpos+i*SHAPE_SIZE);
                ObjectSetInteger(0,name,OBJPROP_YDISTANCE,m_ypos);
            };
        };
        else
        {
            ///--- senkrechter Stock
            for(i=0; i<4; i++)
            {
                name=SHAPE_NAME+(string)i;
                ObjectSetInteger(0,name,OBJPROP_XDISTANCE,m_xpos);
                ObjectSetInteger(0,name,OBJPROP_YDISTANCE,m_ypos+i*SHAPE_SIZE);
            };
        };
    };
};

```

Figur Quadrat ist von der Klasse CTetrisShape6 beschrieben und hat ihre eigene Realisierung der Methode Draw():

```

class CTetrisShape6 : public CTetrisShape
{
public:

```

```

//--- Figurzeichnung
virtual void      Draw()
{
    int      i;
    string name;
    //---
    for(i=0; i<2; i++)
    {
        name=SHAPE_NAME+(string)i;
        ObjectSetInteger(0,name,OBJPROP_XDISTANCE,m_xpos+i*SHAPE_SIZE);
        ObjectSetInteger(0,name,OBJPROP_YDISTANCE,m_ypos);
    };
    for(i=2; i<4; i++)
    {
        name=SHAPE_NAME+(string)i;
        ObjectSetInteger(0,name,OBJPROP_XDISTANCE,m_xpos+(i-2)*SHAPE_SIZE);
        ObjectSetInteger(0,name,OBJPROP_YDISTANCE,m_ypos+SHAPE_SIZE);
    };
};
};
};

```

Abhängig davon, Objekt welcher Klasse erzeugt wird, wird virtuelle Funktion einer oder anderer sekundären Klasse aufgerufen.

```

void CTetrisField::NewShape()
{
    //--- zufallsmaessig schaffen wir eine der 7 moeglichen Figuren
    int nshape=rand()%7;
    switch(nshape)
    {
        case 0: m_shape=new CTetrisShape1; break;
        case 1: m_shape=new CTetrisShape2; break;
        case 2: m_shape=new CTetrisShape3; break;
        case 3: m_shape=new CTetrisShape4; break;
        case 4: m_shape=new CTetrisShape5; break;
        case 5: m_shape=new CTetrisShape6; break;
        case 6: m_shape=new CTetrisShape7; break;
    };
    //--- zeichnen wir
    m_shape.Draw();
    //---
};

```

Sehen Sie auch

[Standardbibliothek](#)

## Statische Mitglieder der Klasse/Struktur

### Statische Mitglieder

Die Mitglieder der Klasse können mit der Storage-Klasse Modifier [statisch](#) deklariert werden. Diese Datenmitglieder werden von allen Instanzen dieser Klasse geteilt und sind an einem Ort gespeichert. Nicht-statische Datenmitglieder werden für jede Variable-Klassenobject erstellt.

Die Unfähigkeit, statische Mitglieder einer Klasse zu deklarieren, würde auf die Notwendigkeit, diese Daten auf der [globalen Ebene](#) des Programms zu deklarieren, geführt haben. Es würde die Beziehung zwischen den Daten und ihrer Klasse brechen und steht nicht im Einklang mit der grundlegenden Paradigma der OOP - die Zusammenführung von Daten und Methoden für Verarbeitung in einer Klasse. Ein statischer Mitglied ermöglicht Klassendaten, die auf eine bestimmte Instanz nicht spezifisch sind, im Gültigkeitsbereich der Klasse zu existieren.

Da eine statische Klasse Mitglied nicht von der jeweiligen Instanz abhängen, ist der Verweis auf ihn wie folgt:

```
class_name::variable
```

wo *class\_name* - ist der Name der Klasse, *variable* ist der name des Mitglieds der Klasse.

Ein statisches Klassenmitglied muss explizit mit den gewünschten Wert initialisiert werden. Um dies zu tun, muss es auf globaler Ebene deklariert und initialisiert werden. Die Reihenfolge der Initialisierung von statischen Mitglieder entspricht der Reihenfolge, in der sie auf der globalen Ebene im Quellcode deklariert sind.

Zum Beispiel haben wir eine Klasse *CParser*, die zum Parsen der Text verwendet wird, und wir brauchen die Gesamtzahl der verarbeiteten Wörter und Zeichen zu zählen. Wir müssen nur die notwendigen Klassenmitglieder als statische deklarieren und sie auf globaler Ebene initialisieren. Dann werden alle Instanzen der Klasse den Zähler von Wörtern und Zeichen gemeinsam nutzen.

```
//+-----+
//| Klasse "Text Parser" |
//+-----+
class CParser
{
public:
    static int      s_words;
    static int      s_symbols;
    //-- Konstruktor und Destruktor
    Parser(void);
    ~Parser(void) {};
};
...
/-- Initialisierung von statischen Mitglieder der Klasse Parser auf globaler Ebene
int CParser::s_words=0;
int CParser::s_symbols=0;
```

Ein statisches Klassenmitglied kann mit dem Schlüsselwort *const* deklariert werden. Solche statischen Konstanten müssen auf globaler Ebene mit dem Schlüsselwort *const* *initialisiert werden*:

```
//+-----+
//| Klasse "Stack" zur Speicherung der Daten, die verarbeitet werden |
//+-----+
class CStack
{
public:
    CStack(void);
    ~CStack(void){};

...
private:
    static const int    s_max_length; // Maximale Kapazität des Stacks
};

//--- initialisierung der statischen Konstanten der Klasse CStack
const int CStack::s_max_length=1000;
```

## Zeiger this

Das Schlüsselwort [this](#) bedeutet implizit deklarierten [Zeiger](#) auf sich selbst - auf eine bestimmte Instanz der Klasse, in deren Rahmen die Methode ausgeführt wird. Es kann nur in nicht-statische Methoden der Klasse verwendet werden. Zeiger this ist ein implizites nicht-statisches Mitglied einer beliebigen Klasse.

In statischen Funktionen können Sie nur statische Mitglieder/Methoden einer Klasse zugreifen.

## Statische Methoden

In MQL5 Mitgliederfunktionen vom Typ [static](#) können verwendet werden. Modifikator *static* muss vor dem Rückgabotyp der Funktion in der Deklaration innerhalb einer Klasse gehen.

```
class CStack
{
public:
    //--- Konstruktor und Destruktor
    CStack(void){};
    ~CStack(void){};

    //--- Maximale Kapazität des Stacks
    static int    Capacity();
private:
    int            m_length;        // Anzahl der Elemente im Stack
    static const int    s_max_length; // Maximale Kapazität des Stacks
};

//+-----+
//| Gibt maximale Anzahl der Elemente im Stack zu speichern zurück |
//+-----+
int CStack::Capacity(void)
{
    return(s_max_length);
};

//--- initialisierung der statischen Konstanten der Klasse CStack
const int CStack::s_max_length=1000;
//+-----+
```

```
///| Script program start function |
//+-----+
void OnStart()
{
//--- eine Variable vom Typ CStack deklarieren
    CStack stack;
//--- die statische Objektmethode anrufen
    Print("CStack.s_max_length=", stack.Capacity());
//--- es ist auch möglich so anzurufen, da die Methode statisch ist und kein Objekt es
    Print("CStack.s_max_length=", CStack::Capacity());
};
```

Eine Methode mit Modifikator **const** heißt konstant und kann nicht implizite Mitglieder ihrer Klasse ändern. Die Deklaration der konstanten Funktionen einer Klasse und konstanten Parametern ist *Kontrolle der Konstanz* (const-correctness). Durch diese Kontrolle können Sie sicher sein, dass der Compiler die Konsistenz der Werte der Objekte gewährleistet und wird einen Fehler während der Kompilierung zurückgeben, wenn etwas falsch ist.

Modifikator **const** wird nach der Liste von Argumenten innerhalb einer Klassendeklaration platziert. Definition außerhalb einer Klasse sollte auch die Modifikator *const* haben:

```

//+-----+
//| Klasse "Rechteck" |
//+-----+
class CRectangle
{
private:
    double      m_width;      // Breite
    double      m_height;     // Höhe
public:
    //--- Konstruktoren und Destruktor
    CRectangle(void):m_width(0),m_height(0){};
    CRectangle(const double w,const double h):m_width(w),m_height(h){};
    ~CRectangle(void){};

    //--- Berechnung der Fläche
    double      Square(void) const;
    static double Square(const double w,const double h); // { return(w*h); }
};

//+-----+
//| Gibt die Fläche des Objekts "Rechteck" zurück |
//+-----+
double CRectangle::Square(void) const
{
    return(Square(m_width,m_height));
}

//+-----+
//| Gibt das Produkt von zwei Variablen zurück |
//+-----+
static double CRectangle::Square(const double w,const double h)
{
    return(w*h);
}

//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    //--- erstellen wir ein Rechteck rect mit Seiten 5 und 6
    CRectangle rect(5,6);
    //--- finden wir Fläche eines Rechtecks mit der konstanten Methode
    PrintFormat("rect.Square()=%.2f",rect.Square());
    //--- finden wir das Produkt der Zahlen mit der statischen Methode der CRectangle-Klasse
    PrintFormat("CRectangle::Square(2.0,1.5)=%f",CRectangle::Square(2.0,1.5));
}

```

Ein zusätzliches Argument für die Verwendung der Konstanz Kontrolle ist die Tatsache, dass in diesem Fall der Compiler eine spezielle Optimierung erzeugt, zum Beispiel, hat ein konstantes Objekt im Nurlesespeicher.

Eine statische Funktion kann nicht mit einem Modifikator `const` definiert werden, da dies Modifikator die Unveränderlichkeit von den Mitgliedern der Instanz garantiert, wenn diese Funktion aufgerufen wird. Aber, wie oben erwähnt, kann die statische Funktion nicht-statische Mitgliedern der Klasse nicht zugreifen.

Sehen Sie auch

[Statische Variablen](#), [Variablen](#), [Referenzen](#). Modifikator `&` und Schlüsselwort `this`



## Funktionsschablonen

Überladene Funktionen werden häufig verwendet, um ähnliche Operationen auf verschiedene Datentypen durchzuführen. Ein einfaches Beispiel für eine solche Funktion im MQL5 ist [ArraySize\(\)](#), die die Größe des Arrays jeder Art zurück gibt. In der Tat ist dieses System Funktion überladen, und die ganze Umsetzung dieses Überladen wird von der MQL5 Software-Entwickler versteckt:

```
int ArraySize(  
    void& array[]    // Array zu prüfen  
);
```

MQL5 Compiler MQL5 ersetzt eine rechte Umsetzung für jeden Aufruf dieser Funktion. Zum Beispiel für Arrays von Typ integer:

```
int ArraySize(  
    int& array[]      // Array mit Elementen vom Typ int  
);
```

Und für einem Array vom Typ [MqlRates](#) für Arbeit mit Preise im Format von historischen Daten kann Funktion [ArraySize\(\)](#) wie folgt dargestellt werden:

```
int ArraySize(  
    MqlRates& array[] // Array mit Elementen vom Typ MqlRates  
);
```

Somit ist es sehr praktisch, die gleiche Funktion für verschiedenen Typen zu nutzen, aber Sie müssen Vorarbeiten ausführen - nämlich Überladen der Funktion für alle Datentypen, mit denen sie arbeitet.

Es gibt eine bequeme Lösung - wenn für jeden Datentyp identische Operationen durchgeführt werden sollte, gibt es eine kompakte und praktische Lösung, Funktionsschablonen zu verwenden. In diesem Fall muss der Programmierer nur eine Beschreibung der Funktionsschablone zu schreiben. Für diese Beschreibung der Schablone, müssen Sie nicht einen spezifischen Datentyp, mit dem die Funktion arbeiten soll, als Parameter eingeben, sondern einen formalen Parameter. Basierend auf der Typen der verwendeten Argumente beim Aufruf der Funktion, wird der Compiler automatisch verschiedene Funktionen für die geeignete Behandlung für jeden Typ generieren.

Die Definition einer Funktionsschablone beginnt mit dem Schlüsselwort [template](#) wonach in spitzen Klammern ist eine Liste der formalen Parameter. Jeder formale Parameter wird durch das Schlüsselwort [typename](#) vorangestellt. Die formalen Parameter-Typen sind eingebaute Typen oder benutzerdefinierte Typen. Sie werden verwendet:

- um die Type der Argumente der Funktion einzugeben
- um den Typ des Rückgabewerts der Funktion einzugeben
- um Variablen innerhalb des Körpers der Beschreibung der Funktion zu erklären

Die Anzahl der Parameter in einer Schablone kann nicht mehr als acht sein. Jeder formale Parameter in der Schablone-Definition sollte mindestens einmal in der Funktion Parameter-Liste erscheinen. Jeder Name vom formalen Parameter muss eindeutig sein.

Hier ist ein Beispiel für ein Funktionsschablone für die Suche nach den maximalen Wert in einem Array von beliebigen numerischen Typs (integer und reelle Zahlen):

```

template<typename T>
T ArrayMax(T &arr[])
{
    uint size=ArraySize(arr);
    if(size==0) return(0);

    T max=arr[0];
    for(uint n=1;n<size;n++)
        if(max<arr[n]) max=arr[n];
    //---
    return(max);
}

```

Dieses Beispiel beschreibt eine Funktion, die den größten Wert in dem übergebenen Array findet, um es als Ergebnis zurück zu geben. Daran erinnern, dass die eingebaute MQL5 Funktion [ArrayMaximum\(\)](#) gibt nur der Index des gefundenen Maximalwert zurück, mit dem der Benutzer kann in der Zukunft den Wert selbst bekommen. Zum Beispiel:

```

//--- Erstellen wir ein Array
double array[];
int size=50;
ArrayResize(array,size);
//--- Füllen es mit zufälligen Werten
for(int i=0;i<size;i++)
{
    array[i]=MathRand();
}

//--- Finden wir die Position der maximalen Element im Array
int max_position=ArrayMaximum(array);
//--- Bekommen wir das Maximalwert im Array
double max=array[max_position];
//--- Ausgabe des gefundenen Wertes
Print("Max value = ",max);

```

So brauchen wir zwei Schritte, um den maximalen Wert in dem Array zu erhalten. Mit Hilfe der Funktionsschablone `ArrayMax()` kann man das Ergebnis vom gewünschten Typ erhalten, wenn man ein Array des entsprechenden Typs in sie übergibt. Das heißt, anstelle der zwei letzten Zeilen

```

//--- Finden wir die Position der maximalen Element im Array
int max_position=ArrayMaximum(array);
//--- Bekommen wir das Maximalwert im Array
double max=array[max_position];

```

Jetzt können wir nur eine einzige Zeile, die ein Ergebnis des gleichen Typs wie der übergebenen Array zurückgibt, benutzen:

```

//--- Finden wir den maximalen Wert
double max=ArrayMax(array);

```

Der Typ des Ergebnisses, der durch `ArrayMax()` zurückgegeben wird, wird automatisch mit dem Typ des Arrays gleich sein.

Um mit verschiedenen Typen von Daten zu arbeiten, müssen Sie das Schlüsselwort `typename` verwenden, um den Typ des Arguments als Zeichen zu erhalten. Wir zeigen dies am Beispiel einer Funktion, die einen Datentyp als String zurückgibt:

```

#include <Trade\Trade.mqh>
//+-----+
//|                                     |
//+-----+
void OnStart()
{
//---
    CTrade trade;
    double d_value=M_PI;
    int i_value=INT_MAX;
    Print("d_value: type=",GetTypeName(d_value), ",   value=", d_value);
    Print("i_value: type=",GetTypeName(i_value), ",   value=", i_value);
    Print("trade: type=",GetTypeName(trade));
//---
}
//+-----+
//| Gibt den Typ als String zurück      |
//+-----+
template<typename T>
string GetTypeName(const T &t)
{
//--- den Typ als String zurückgeben
    return(GetTypeName(T));
//---
}

```

Funktionsschablonen können auch für Klassenmethode verwendet werden, zum Beispiel:

```

class CFile
{
    ...
public:
    ...
    template<typename T>
    uint WriteStruct(T &data);
};

template<typename T>
uint CFile::WriteStruct(T &data)
{
    ...
    return(FileWriteStruct(m_handle,data));
}

```

Funktionsschablonen können nicht mit den Schlüsselwörtern [export](#), [virtual](#) und [#import](#) deklariert werden.

## Standardkonstanten, Aufzählungen und Strukturen

für die Erleichterung von Programmschreiben und gute Wahrnehmung von Ausgangsprogrammtexten werden in der Sprache MQL5 vorbestimmte Standardkonstanten und Aufzählungen vorausgesehen. Außerdem werden für Informationsspeicherung dienstliche [Strukturen](#) verwendet.

Standardkonstanten sind Analoge der Makrosubstitutionen und haben den Typ [int](#).

Konstanten sind nach ihrem Zweck gruppiert:

- [Chartkonstanten](#) werden bei der Arbeit mit Preischarts benutzt: Öffnung, Navigation, Parametereinstellung;
- [Objektkonstanten](#) sind für die Verarbeitung graphischer Objekte bestimmt, die erstellt und in Charten dargestellt werden können;
- [Indikatorkonstanten](#) dienen der Arbeit mit Standard- und Benutzerindikatoren;
- [Medium\\_Zustand](#) - beschreiben Eigenschaften eines mql5-Programms, geben Information über das Kundenterminal, Handelsinstrument und das laufende Handelskonto;
- [Handelskonstanten](#) ermöglichen es verschiedenartige Information während Handelsprozesses zu praezisieren ;
- [Benannte Konstanten](#) - Konstanten der Sprache MQL5;
- [Datenstrukturen](#) beschreiben Verwendete Formate der Datenspeicherung;
- [Fehler- Warnungskoden](#) beschreiben Compilernachrichten und Nachrichten des Handelsservers auf Handelsanforderungen
- [Input/Output Konstanten](#) sind für die Arbeit mit [Dateifunktionen](#) und Ausgabe von Informationen auf Computerdislay durch die Funktion [MessageBox\(\)](#) bestimmt.

## Chartkonstanten

Konstanten, die verschiedenen Charteigenschaften beschreiben sind in folgende Gruppen aufgeteilt:

- [Typen der Ereignisse](#) - Ereignisse, die bei der Arbeit mit Chart vorkommen;
- [Chartperioden](#) - eingebaute Standardperioden;
- [Charteigenschaften](#) - Identifikatoren, die als Parameter der [Funktionen für Arbeit mit Charts](#) verwendet werden;
- [Positionierungskonstanten](#) - Parameterwerte der Funktion [ChartNavigate\(\)](#);
- [Darstellung von Charts](#) - Einsetzung von Chartaussehen.

## Typen der Chartereignisse

Es gibt 9 Typen der Ereignisse, die durch die vorbestimmte Funktion [OnChartEvent\(\)](#) bearbeitet werden können. Für Benutzerereignisse werden 65535 Identifikatoren im Bereich von CHARTEVENT\_CUSTOM bis CHARTEVENT\_CUSTOM\_LAST vorausgesehen. Für Generieren des Benutzerereignisses muss man die Funktion [EventChartCustom\(\)](#) verwenden.

### ENUM\_CHART\_EVENT

| Identifikator             | Beschreibung                                                                                                                                         |
|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| CHARTEVENT_KEYDOWN        | Tasten                                                                                                                                               |
| CHARTEVENT_MOUSE_MOVE     | Das Bewegen der Maus und Mausklicks (wenn die Eigenschaft <a href="#">CHART_EVENT_MOUSE_MOVE=true</a> ist für den Chart eingegeben)                  |
| CHARTEVENT_OBJECT_CREATE  | Erzeugung vom <a href="#">graphischen Objekt</a> (wenn die Eigenschaft <a href="#">CHART_EVENT_OBJECT_CREATE=true</a> ist für den Chart eingegeben)  |
| CHARTEVENT_OBJECT_CHANGE  | Veränderung der Objekteigenschaften durch Dialog der Eigenschaften                                                                                   |
| CHARTEVENT_OBJECT_DELETE  | Entfernung vom <a href="#">graphischen Objekt</a> (wenn die Eigenschaft <a href="#">CHART_EVENT_OBJECT_DELETE=true</a> ist für den Chart eingegeben) |
| CHARTEVENT_OBJECT_CLICK   | Betaetigen der Taste mit der Maus auf dem <a href="#">graphischen Objekt</a>                                                                         |
| CHARTEVENT_OBJECT_DRAG    | Ziehen des <a href="#">graphischen Objektes</a>                                                                                                      |
| CHARTEVENT_OBJECT_ENDEDIT | Beenden von Texteditieren im graphischen Objekt Edit                                                                                                 |
| CHARTEVENT_CHART_CHANGE   | Ereignis der Veränderung der Chart-Größe oder Veränderung der Chart-Eigenschaften durch Dialog von Eigenschaften                                     |
| CHARTEVENT_CUSTOM         | Anfangsnummer des Ereignisses vom Bereich der Benutzerereignisse                                                                                     |
| CHARTEVENT_CUSTOM_LAST    | Endnummer des Ereignisses vom Bereich der Benutzerereignisse.                                                                                        |

Für jeden Typ des Ereignisses haben Eingabeparameter der Funktion OnChartEvent() vorbestimmte Werte, die für Verarbeitung dieses Ereignisses notwendig sind. In der Tabelle sind Ereignisse und Werte aufgezählt, die durch Parameter übertragen werden.

| Ereignis | Wert | des | Wert | des | Wert | des | Wert | des |
|----------|------|-----|------|-----|------|-----|------|-----|
|----------|------|-----|------|-----|------|-----|------|-----|

|                                                                                                                                                      | Parameters id            | Parameters lparam         | Parameters dparam                                                                                   | Parameters sparam                                                              |
|------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------|---------------------------|-----------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------|
| Ereignis des Keyboard Klickens                                                                                                                       | CHARTEVENT_KEYDOWN       | Kode der betätigten Taste | Die Anzahl der Tastenanschläge, die generiert werden, während diese Taste in gedrücktem Zustand war | Der String-Wert eines Bit-Maske, die den Status der Tastatur-Tasten beschreibt |
| Mausereignisse (wenn die Eigenschaft <a href="#">CHART_EVENT_MOUSE_MOVE</a> =true ist für den Chart eingegeben)                                      | CHARTEVENT_MOUSE_MOVE    | X-Koordinate              | Y-Koordinate                                                                                        | Der String-Wert einer Bit-Maske, die den Status der Maustasten beschreibt      |
| Ereignis der Erzeugung des graphischen Objektes (wenn die Eigenschaft <a href="#">CHART_EVENT_OBJECT_CREATE</a> =true ist für den Chart eingegeben)  | CHARTEVENT_OBJECT_CREATE | —                         | —                                                                                                   | Name des erzeugten graphischen Objektes                                        |
| Ereignis der Veränderung der Eigenschaften des Objektes durch Dialog der Eigenschaften                                                               | CHARTEVENT_OBJECT_CHANGE | —                         | —                                                                                                   | Name des veränderten graphischen Objektes                                      |
| Ereignis der Entfernung des graphischen Objektes (wenn die Eigenschaft <a href="#">CHART_EVENT_OBJECT_DELETE</a> =true ist für den Chart eingegeben) | CHARTEVENT_OBJECT_DELETE | —                         | —                                                                                                   | Name des entfernten graphischen Objektes                                       |
| Ereignis des Mausklicks auf der graphischen Darstellung                                                                                              | CHARTEVENT_CLICK         | X-Koordinate              | Y-Koordinate                                                                                        | —                                                                              |

|                                                                                                                  |                           |                                                                        |                                                                        |                                                                                        |
|------------------------------------------------------------------------------------------------------------------|---------------------------|------------------------------------------------------------------------|------------------------------------------------------------------------|----------------------------------------------------------------------------------------|
| Ereignis des Mausklicks auf dem graphischen Objekt                                                               | CHARTEVENT_OBJECT_CLICK   | X-Koordinate                                                           | Y-Koordinate                                                           | Name des graphischen Objektes, auf dem das Ereignis geschehen ist                      |
| Ereignis der Verschiebung des graphischen Objektes per Mausklick                                                 | CHARTEVENT_OBJECT_DRAG    | —                                                                      | —                                                                      | Name des verschobenen graphischen Objektes                                             |
| Ereignis der Beendung des Texteditierens im Eingabefeld des graphischen Objektes "Eingabefeld"                   | CHARTEVENT_OBJECT_ENDEDIT | —                                                                      | —                                                                      | Name des graphischen Objektes "Eingabefeld", in dem Editieren des Textes beendet wurde |
| Ereignis der Veränderung der Chart-Größe oder Veränderung der Chart-Eigenschaften durch Dialog von Eigenschaften | CHARTEVENT_CHART_CHANGE   | —                                                                      | —                                                                      | —                                                                                      |
| Benutzerereignis mit Nummer N                                                                                    | CHARTEVENT_CUSTOM+N       | Wert, vorgegeben durch die Funktion <a href="#">EventChartCustom()</a> | Wert, vorgegeben durch die Funktion <a href="#">EventChartCustom()</a> | Wert, vorgegeben durch die Funktion <a href="#">EventChartCustom()</a>                 |

Beispiel:



```
#define KEY_NUMPAD_5      12
#define KEY_LEFT         37
#define KEY_UP           38
#define KEY_RIGHT        39
#define KEY_DOWN         40
#define KEY_NUMLOCK_DOWN 98
#define KEY_NUMLOCK_LEFT 100
#define KEY_NUMLOCK_5    101
#define KEY_NUMLOCK_RIGHT 102
#define KEY_NUMLOCK_UP   104

//+-----+
//| Expert initialization function |
//+-----+
int OnInit()
{
    //---
    Print("The expert with name ",MQL5InfoString(MQL5_PROGRAM_NAME)," is running");
    //--- enable object create events
    ChartSetInteger(ChartID(),CHART_EVENT_OBJECT_CREATE,true);
    //--- enable object delete events
```

```

    ChartSetInteger(ChartID(), CHART_EVENT_OBJECT_DELETE, true);
//---
    return(INIT_SUCCEEDED);
}
//+-----+
//|                                           |
//+-----+
void OnChartEvent(const int id,          // Event identifier
                  const long& lparam,    // Event parameter of long type
                  const double& dparam,  // Event parameter of double type
                  const string& sparam   // Event parameter of string type
                  )
{
//--- the left mouse button has been pressed on the chart
    if(id==CHARTEVENT_CLICK)
    {
        Print("The coordinates of the mouse click on the chart are: x = ", lparam, " y = ");
    }
//--- the mouse has been clicked on the graphic object
    if(id==CHARTEVENT_OBJECT_CLICK)
    {
        Print("The mouse has been clicked on the object with name '"+sparam+"'");
    }
//--- the key has been pressed
    if(id==CHARTEVENT_KEYDOWN)
    {
        switch(lparam)
        {
            case KEY_NUMLOCK_LEFT:  Print("The KEY_NUMLOCK_LEFT has been pressed"); break;
            case KEY_LEFT:          Print("The KEY_LEFT has been pressed");       break;
            case KEY_NUMLOCK_UP:    Print("The KEY_NUMLOCK_UP has been pressed");   break;
            case KEY_UP:            Print("The KEY_UP has been pressed");           break;
            case KEY_NUMLOCK_RIGHT: Print("The KEY_NUMLOCK_RIGHT has been pressed"); break;
            case KEY_RIGHT:         Print("The KEY_RIGHT has been pressed");        break;
            case KEY_NUMLOCK_DOWN:  Print("The KEY_NUMLOCK_DOWN has been pressed");  break;
            case KEY_DOWN:          Print("The KEY_DOWN has been pressed");         break;
            case KEY_NUMPAD_5:      Print("The KEY_NUMPAD_5 has been pressed");     break;
            case KEY_NUMLOCK_5:    Print("The KEY_NUMLOCK_5 has been pressed");     break;
            default:                Print("Some not listed key has been pressed");
        }
        ChartRedraw();
    }
//--- the object has been deleted
    if(id==CHARTEVENT_OBJECT_DELETE)
    {
        Print("The object with name ", sparam, " has been deleted");
    }
//--- the object has been created
    if(id==CHARTEVENT_OBJECT_CREATE)
    {
        Print("The object with name ", sparam, " has been created");
    }
//--- the object has been moved or its anchor point coordinates has been changed
    if(id==CHARTEVENT_OBJECT_DRAG)
    {
        Print("The anchor point coordinates of the object with name ", sparam, " has been changed");
    }
//--- the text in the Edit of object has been changed
    if(id==CHARTEVENT_OBJECT_ENDEDIT)
    {
        Print("The text in the Edit field of the object with name ", sparam, " has been changed");
    }
}

```

```
    }  
}
```

Для события CHARTEVENT\_MOUSE\_MOVE строковой параметр `sparam` содержит число, представляющее информацию о состоянии клавиш:

| Бит | Описание                                     |
|-----|----------------------------------------------|
| 1   | Состояние левой клавиши мыши                 |
| 2   | Для MQL5-программ всегда ноль                |
| 3   | Состояние клавиши SHIFT                      |
| 4   | Состояние клавиши CTRL                       |
| 5   | Состояние средней клавиши мыши               |
| 6   | Состояние первой дополнительной клавиши мыши |
| 7   | Состояние второй дополнительной клавиши мыши |

Пример:

```
//+-----+  
//| Expert initialization function |  
//+-----+  
void OnInit()  
{  
    //--- включение сообщений о перемещении мыши по окну чарта  
    ChartSetInteger(0, CHART_EVENT_MOUSE_MOVE, 1);  
}  
//+-----+  
//| MouseState |  
//+-----+  
string MouseState(uint state)  
{  
    string res;  
    res+="\nML: " + (((state & 1) == 1) ? "DN": "UP"); // mouse left  
    res+="\nMR: " + (((state & 2) == 2) ? "DN": "UP"); // mouse right (always zero)  
    res+="\nMM: " + (((state & 16) == 16) ? "DN": "UP"); // mouse middle  
    res+="\nMX: " + (((state & 32) == 32) ? "DN": "UP"); // mouse first X key  
    res+="\nMY: " + (((state & 64) == 64) ? "DN": "UP"); // mouse second X key  
    res+="\nSHIFT: " + (((state & 4) == 4) ? "DN": "UP"); // shift key  
    res+="\nCTRL: " + (((state & 8) == 8) ? "DN": "UP"); // control key  
    return(res);  
}  
//+-----+  
//| ChartEvent function |  
//+-----+  
void OnChartEvent(const int id, const long &lparam, const double &dparam, const string &sparam)
```

```
{  
    if(id==CHARTEVENT_MOUSE_MOVE)  
        Comment("POINT: ",(int)lparam,",",(int)dparam,"\n",MouseState((uint)sparam));  
}
```

Sehen Sie auch

[Funktionen der Bearbeitung von Ereignissen](#), [Arbeit mit Ereignissen](#)

## Chartperioden

Alle vorbestimmten Programme haben unikale Identifikatoren. Identifikator `PERIOD_CURRENT` bezeichnet laufende Periode des Charts, in dem das mql5-Programm aktiviert ist .

### ENUM\_TIMEFRAMES

| Identifikator  | Beschreibung     |
|----------------|------------------|
| PERIOD_CURRENT | laufende Periode |
| PERIOD_M1      | 1 Minute         |
| PERIOD_M2      | 2 Minuten        |
| PERIOD_M3      | 3 Minuten        |
| PERIOD_M4      | 4 Minuten        |
| PERIOD_M5      | 5 Minuten        |
| PERIOD_M6      | 6 Minuten        |
| PERIOD_M10     | 10 Minuten       |
| PERIOD_M12     | 12 Minuten       |
| PERIOD_M15     | 15 Minuten       |
| PERIOD_M20     | 20 Minuten       |
| PERIOD_M30     | 30 Minuten       |
| PERIOD_H1      | 1 Stunde         |
| PERIOD_H2      | 2 Stunden        |
| PERIOD_H3      | 3 Stunden        |
| PERIOD_H4      | 4 Stunden        |
| PERIOD_H6      | 6 Stunden        |
| PERIOD_H8      | 8 Stunden        |
| PERIOD_H12     | 12 Stunden       |
| PERIOD_D1      | 1 Tag            |
| PERIOD_W1      | 1 Woche          |
| PERIOD_MN1     | 1 Monat          |

### Beispiel:

```
string chart_name="test_Object_Chart";
Print("Versuchen wir das Objekt Chart zu erzeugen mit dem Namen ",chart_name);
//--- wenn dieses Objekt fehlt, schaffen wir das
if(ObjectFind(0,chart_name)<0)ObjectCreate(0,chart_name,OBJ_CHART,0,0,0,0,0);
```

```
//--- geben wir das Symbol vor
    ObjectSetString(0,chart_name,OBJPROP_SYMBOL,"EURUSD");
//--- geben wir X-Koordinate für Bezugspunkt vor
    ObjectSetInteger(0,chart_name,OBJPROP_XDISTANCE,100);
//--- geben wir Y-Koordinate für Bezugspunkt vor
    ObjectSetInteger(0,chart_name,OBJPROP_YDISTANCE,100);
//--- stellen wir die Breite ein
    ObjectSetInteger(0,chart_name,OBJPROP_XSIZE,400);
//--- stellen wir die Hoehe ein
    ObjectSetInteger(0,chart_name,OBJPROP_YSIZE,300);
//--- stellen wir die Periode ein
    ObjectSetInteger(0,chart_name,OBJPROP_PERIOD,PERIOD_D1);
//--- Stellen wir Massstab ein ( von 0 bis 5)
    ObjectSetDouble(0,chart_name,OBJPROP_SCALE,4);
//--- machen wir es unzugänglich für Hervorheben mit der Maus
    ObjectSetInteger(0,chart_name,OBJPROP_SELECTABLE,false);
```

Sehen Sie auch

[PeriodSeconds](#), [Period](#), [Datum und Zeit](#), [Objektsichtbarkeit](#)

## Charteigenschaften

Identifikatoren der Aufzählungsfamilie ENUM\_CHART\_PROPERTY verwenden als Parameter [Funktionen für Arbeit mit Charts](#). Initialkurzwort r/o in der Spalte "Typ der Eigenschaft" bedeutet, dass diese Eigenschaft erst für Lesen bestimmt ist und kann nicht verändert werden. Abkürzung w/o in der Spalte "Typ der Eigenschaft" bedeutet, dass diese Eigenschaft write-only ist und kann nicht erhalten werden. Beim Zugriff zu einigen Eigenschaften ist es notwendig, zusätzlichen Parameter-Modifikator (modifier) anzugeben, der der Andeutung der Nummer von Chartsubfenster dient.

Funktionen, die Eigenschaften des Charts setzen, werden in der Tat verwendet, um Befehle zu ändern zu senden. Der erfolgreiche Erfüllung dieser Funktionen wird der Befehl in die allgemeine Warteschlange der Chartereignisse platziert. Der Chart wird im Zuge der Verarbeitung der Ereigniswarteschlange von diesem Chart geändert.

Aus diesem Grund sollte man nicht erwarten, dass der Chart sofort nach dem Aufruf dieser Funktionen visuell aktualisiert wird. Im Allgemeinen ist der Chart durch das Terminal automatisch nach Änderung-Ereignisse aktualisiert - die Ankunft der neuen Quote, Änderung von Chartfenstergröße, etc.

Um Aussehen des Charts zu aktualisieren, verwenden Sie [ChartRedraw\(\)](#).

für Funktionen [ChartSetInteger\(\)](#) und [ChartGetInteger\(\)](#)

### ENUM\_CHART\_PROPERTY\_INTEGER

| Identifikator                          | Beschreibung                                                                                                                                                                                                            | Typ der Eigenschaft |
|----------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------|
| <a href="#">CHART_IS_OBJECT</a>        | Attribut um Objekt "Chart" ( <a href="#">OBJ_CHART</a> ) zu identifizieren - für ein grafisches Objekt gibt true zurück. Für dieses Chart ist die Eigenschaft false                                                     | bool r/o            |
| <a href="#">CHART_BRING_TO_TOP</a>     | Den Chart über anderen Chart anzeigen                                                                                                                                                                                   | bool w/o            |
| <a href="#">CHART_MOUSE_SCROLL</a>     | Den Chart mit der linken Maustaste horizontal scrollen. Vertikales Scrollen wird verfügbar sein, wenn der Wert von einer der drei Eigenschaften true ist: CHART_SCALEFIX, CHART_SCALEFIX_11 oder CHART_SCALE_PT_PER_BAR | bool                |
| <a href="#">CHART_EVENT_MOUSE_MOVE</a> | Senden alle MQL5-Programme auf dem Chart Benachrichtigungen über Ereignisse der Bewegung und Klicken der Maustasten ( <a href="#">CHARTEVENT_MOUSE_MOVE</a> )                                                           | bool                |

|                                                  |                                                                                                                                                               |                                             |
|--------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------|
| <a href="#"><u>CHART_EVENT_OBJECT_CREATE</u></a> | Senden alle MQL5-Programme auf dem Chart eine Benachrichtigung über Ereignis des neuen Objekts Erstellung ( <a href="#"><u>CHARTEVENT_OBJECT_CREATE</u></a> ) | bool                                        |
| <a href="#"><u>CHART_EVENT_OBJECT_DELETE</u></a> | Senden alle MQL5-Programme auf dem Chart eine Benachrichtigung über Ereignis des Objekts Löschung ( <a href="#"><u>CHARTEVENT_OBJECT_DELETE</u></a> )         | bool                                        |
| <a href="#"><u>CHART_MODE</u></a>                | Charttyp (Kerzen, Bars, oder Linie)                                                                                                                           | enum <a href="#"><u>ENUM_CHART_MODE</u></a> |
| <a href="#"><u>CHART_FOREGROUND</u></a>          | Preischart auf dem Hintergrund                                                                                                                                | bool                                        |
| <a href="#"><u>CHART_SHIFT</u></a>               | Absatz des Preischarts rechtsbündig                                                                                                                           | bool                                        |
| <a href="#"><u>CHART_AUTOSCROLL</u></a>          | Automatischer Sprung zum rechten Schlussrand des Charts                                                                                                       | bool                                        |
| <a href="#"><u>CHART_SCALE</u></a>               | Maßstab                                                                                                                                                       | int von 0 bis 5                             |
| <a href="#"><u>CHART_SCALEFIX</u></a>            | Festmaßstab                                                                                                                                                   | bool                                        |
| <a href="#"><u>CHART_SCALEFIX_11</u></a>         | Maßstab 1:1                                                                                                                                                   | bool                                        |
| <a href="#"><u>CHART_SCALE_PT_PER_BAR</u></a>    | Maßstab in Punkt/Balken                                                                                                                                       | bool                                        |
| <a href="#"><u>CHART_SHOW_OHLC</u></a>           | Darstellung der Werte OHLC im oberen linken Winkel                                                                                                            | bool                                        |
| <a href="#"><u>CHART_SHOW_BID_LINE</u></a>       | Darstellung des Wertes Bid als horizontale Linie auf dem Chart                                                                                                | bool                                        |
| <a href="#"><u>CHART_SHOW_ASK_LINE</u></a>       | Darstellung des Wertes Ask als horizontale Linie auf dem Chart                                                                                                | bool                                        |
| <a href="#"><u>CHART_SHOW_LAST_LINE</u></a>      | Darstellung des Wertes Last durch horizontale Linie auf dem Chart                                                                                             | bool                                        |
| <a href="#"><u>CHART_SHOW_PERIOD_SEP</u></a>     | Darstellung der senkrechten Begrenzungen unter Nachbarperioden                                                                                                | bool                                        |
| <a href="#"><u>CHART_SHOW_GRID</u></a>           | Darstellung des Zeitrasters auf dem Chart                                                                                                                     | bool                                        |



|                                                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                                                       |
|------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------|
| <a href="#"><u>CHART_SHOW_VOLUMES</u></a>      | Darstellung von Volumen auf dem Chart                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | enum<br><a href="#"><u>ENUM_CHART_VOLUME_MODE</u></a> |
| <a href="#"><u>CHART_SHOW_OBJECT_DESCR</u></a> | Popup-Beschreibungen der graphischen Objekte                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | bool                                                  |
| <a href="#"><u>CHART_VISIBLE_BARS</u></a>      | Anzahl der Balken auf dem Chart, zugänglich für die Darstellung                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | int r/o                                               |
| <a href="#"><u>CHART_WINDOWS_TOTAL</u></a>     | Gesamtzahl der Chartfenster, einschließlich Subfenster der Indikatoren                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | int r/o                                               |
| <a href="#"><u>CHART_WINDOW_IS_VISIBLE</u></a> | Sichtbarkeit der Subfenster                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | bool r/o      Modifikator - Nummer des Subfensters    |
| <a href="#"><u>CHART_WINDOW_HANDLE</u></a>     | Handle des Charts (HWND)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | int r/o                                               |
| <a href="#"><u>CHART_WINDOW_YDISTANCE</u></a>  | Der Abstand zwischen dem oberen Rahmen des Unterfensters eines Indikators und dem oberen Rahmen des Hauptfensters des Charts, entlang der vertikalen Y-Achse, in Pixeln. Im Falle eines Maus-Ereignisses, werden die Cursor-Koordinaten im Koordinaten des Hauptfensters übergeben, während die Koordinaten von grafischen Objekten im Indikator-Unterfenster relativ zur oberen linken Ecke des Unterfensters eingestellt sind. Der Wert ist erforderlich, um die absoluten Koordinaten des Hauptfensters zu den lokalen Koordinaten eines Unterfensters für die korrekte Arbeit mit der grafischen Objekte, deren Koordinaten sind relativ zur oberen linken Ecke des Unterfensters angegeben, umzuwandeln. | int r/o      Modifikator - Nummer des Subfensters     |
| <a href="#"><u>CHART_FIRST_VISIBLE_BAR</u></a> | Nummer der ersten sichtbaren Balken auf dem Chart. Indizierung von Balken ist das gleiche wie für <a href="#"><u>Zeitreihen</u></a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | int r/o                                               |
| <a href="#"><u>CHART_WIDTH_IN_BARS</u></a>     | Chartbreite in Balken                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | int r/o                                               |
| <a href="#"><u>CHART_WIDTH_IN_PIXELS</u></a>   | Chartbreite in Pixel                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | int r/o                                               |

|                                |                                                                                                                           |                                          |
|--------------------------------|---------------------------------------------------------------------------------------------------------------------------|------------------------------------------|
| <u>CHART_HEIGHT_IN_PIXELS</u>  | Charthöhe in Pixel                                                                                                        | int Modifikator - Nummer des Subfensters |
| <u>CHART_COLOR_BACKGROUND</u>  | Hintergrundfarbe des Charts                                                                                               | color                                    |
| <u>CHART_COLOR_FOREGROUND</u>  | Farbe von Achsen, Skale und Zeile OHLC                                                                                    | color                                    |
| <u>CHART_COLOR_GRID</u>        | Farbe des Zeitrasters                                                                                                     | color                                    |
| <u>CHART_COLOR_VOLUME</u>      | Farbe der Volumen und Levels der offenen Positionen                                                                       | color                                    |
| <u>CHART_COLOR_CHART_UP</u>    | Farbe des Balkens oben, des Schattens und der Umrandung des Körpers der Bullenkerze                                       | color                                    |
| <u>CHART_COLOR_CHART_DOWN</u>  | Farbe des Balkens unten, des Schattens und Umrandung des Körpers der Bärenkerze                                           | color                                    |
| <u>CHART_COLOR_CHART_LINE</u>  | Farbe der Chartlinie und der japanischen Kerzen "Dodzhi"                                                                  | color                                    |
| <u>CHART_COLOR_CANDLE_BULL</u> | Körperfarbe der Bullenkerze                                                                                               | color                                    |
| <u>CHART_COLOR_CANDLE_BEAR</u> | Körperfarbe der Bärenkerze                                                                                                | color                                    |
| <u>CHART_COLOR_BID</u>         | Linienfarbe des Bid-Preises                                                                                               | color                                    |
| <u>CHART_COLOR_ASK</u>         | Linienfarbe des Ask-Preises                                                                                               | color                                    |
| <u>CHART_COLOR_LAST</u>        | Linienfarbe des Preises des letzten abgeschlossenen Deals (Last)                                                          | color                                    |
| <u>CHART_COLOR_STOP_LEVEL</u>  | Levelsfarbe der Stop-Ordern (Stop Loss und Take Profit)                                                                   | color                                    |
| <u>CHART_SHOW_TRADE_LEVELS</u> | Darstellung der Handelsniveaus auf dem Chart (Niveaus der offenen Positionen, Stop Loss, Take Profit und wartender Order) | bool                                     |
| <u>CHART_DRAG_TRADE_LEVELS</u> | ziehen der Handelsstufen auf dem Chart mit dem Maus erlauben. Standardmäßig ist der Ziehen-Modus aktiviert (true)         | bool                                     |
| <u>CHART_SHOW_DATE_SCALE</u>   | Anzeigen die Zeit-Skala auf dem Diagramm                                                                                  | bool                                     |
| <u>CHART_SHOW_PRICE_SCALE</u>  | Anzeigen die Preis-Skala auf dem Diagramm                                                                                 | bool                                     |

|                                             |                                                                                                         |      |
|---------------------------------------------|---------------------------------------------------------------------------------------------------------|------|
| <a href="#"><u>CHART_SHOW_ONE_CLICK</u></a> | Отображение на графике панели быстрой торговли (опция " <a href="#"><u>Торговля одним кликом</u></a> ") | bool |
|---------------------------------------------|---------------------------------------------------------------------------------------------------------|------|

für die Funktionen [ChartSetDouble\(\)](#) und [ChartGetDouble\(\)](#)

#### ENUM\_CHART\_PROPERTY\_DOUBLE

| Identifikator                               | Beschreibung                                                                                                                                                                                                                                                                                                                                                                                                                                           | Typ der Eigenschaft                             |
|---------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------|
| <a href="#"><u>CHART_SHIFT_SIZE</u></a>     | Absatzgröße der Nullbar vom rechten Schlussrand                                                                                                                                                                                                                                                                                                                                                                                                        | double (von 10 bis 50 Prozent)                  |
| <a href="#"><u>CHART_FIXED_POSITION</u></a> | Position der festen Position des Chart vom linken Rand als Prozentsatz. Feste Position des Charts ist mit einem kleinen grauen Dreieck auf der horizontalen Achse der Zeit gezeichnet und wird nur angezeigt, wenn Autoscroll des Charts mit einem neuen Tick nach rechts deaktiviert ist (siehe Eigenschaft <a href="#"><u>CHART_AUTOSCROLL</u></a> ). Der Bar, der in der feste Position ist, bleibt an der gleiche Stelle mit Zoom-in und Zoom-out. | double                                          |
| <a href="#"><u>CHART_FIXED_MAX</u></a>      | Festes Chartmaximum                                                                                                                                                                                                                                                                                                                                                                                                                                    | double                                          |
| <a href="#"><u>CHART_FIXED_MIN</u></a>      | Festes Chartminimum                                                                                                                                                                                                                                                                                                                                                                                                                                    | double                                          |
| <a href="#"><u>CHART_POINTS_PER_BAR</u></a> | Massstabwert in Punkt/Balken                                                                                                                                                                                                                                                                                                                                                                                                                           | double                                          |
| <a href="#"><u>CHART_PRICE_MIN</u></a>      | Chartminimum                                                                                                                                                                                                                                                                                                                                                                                                                                           | double r/o Modifikator - Nummer des Subfensters |
| <a href="#"><u>CHART_PRICE_MAX</u></a>      | Chartmaximum                                                                                                                                                                                                                                                                                                                                                                                                                                           | double r/o Modifikator - Nummer des Subfensters |

für die Funktionen [ChartSetString\(\)](#) und [ChartGetString\(\)](#)

#### ENUM\_CHART\_PROPERTY\_STRING

| Identifikator                        | Beschreibung                | Typ der Eigenschaft |
|--------------------------------------|-----------------------------|---------------------|
| <a href="#"><u>CHART_COMMENT</u></a> | Kommentartext auf dem Chart | string              |

Beispiel:

```
int chartMode=ChartGetInteger(0,CHART_MODE);
switch(chartMode)
```

```
{
    case(CHART_BARS):    Print("CHART_BARS");    break;
    case(CHART_CANDLES): Print("CHART_CANDLES");break;
    default:Print("CHART_LINE");
};
bool shifted=ChartGetInteger(0,CHART_SHIFT);
if(shifted) Print("CHART_SHIFT = true");
else Print("CHART_SHIFT = false");
bool autoscroll=ChartGetInteger(0,CHART_AUTOSCROLL);
if(autoscroll) Print("CHART_AUTOSCROLL = true");
else Print("CHART_AUTOSCROLL = false");
int chartHandle=ChartGetInteger(0,CHART_WINDOW_HANDLE);
Print("CHART_WINDOW_HANDLE = ",chartHandle);
int windows=ChartGetInteger(0,CHART_WINDOWS_TOTAL);
Print("CHART_WINDOWS_TOTAL = ",windows);
if(windows>1)
{
    for(int i=0;i<windows;i++)
    {
        int height=ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,i);
        double priceMin=ChartGetDouble(0,CHART_PRICE_MIN,i);
        double priceMax=ChartGetDouble(0,CHART_PRICE_MAX,i);
        Print(i+": CHART_HEIGHT_IN_PIXELS = ",height," pixels");
        Print(i+": CHART_PRICE_MIN = ",priceMin);
        Print(i+": CHART_PRICE_MAX = ",priceMax);
    };
};
```

Sehen Sie auch

[Beispiele von Arbeiten mit dem Chart](#)

## Positionieren des Charts

Drei Identifikatoren aus der Liste `ENUM_CHART_POSITION` sind mögliche Werte des Parameters `position` für die Funktion [ChartNavigate\(\)](#).

### ENUM\_CHART\_POSITION

| Identifikator     | Beschreibung                            |
|-------------------|-----------------------------------------|
| CHART_BEGIN       | Anfang des Charts (die ältesten Preise) |
| CHART_CURRENT_POS | Laufende Position                       |
| CHART_END         | Ende des Charts (die neuesten Preise)   |

### Beispiel:

```
long handle=ChartOpen("EURUSD",PERIOD_H12);
if(handle!=0)
{
    ChartSetInteger(handle,CHART_AUTOSCROLL,false);
    ChartSetInteger(handle,CHART_SHIFT,true);
    ChartSetInteger(handle,CHART_MODE,CHART_LINE);
    ResetLastError();
    bool res=ChartNavigate(handle,CHART_END,150);
    if(!res) Print("Navigate failed. Error = ",GetLastError());
    ChartRedraw();
};
```

## Darstellung des Charts

Preischarts können durch drei Verfahren dargestellt werden:

- in Form von Bars;
- in Form von Kerzen;
- in Form von einer gebrochenen Linie.

Konkretes Verfahren der Darstellung vom Preischart ist durch die Funktion [ChartSetInteger](#)(handle\_des\_Diagramms, [CHART\\_MODE](#), Typ\_des\_Charts) eingesetzt, wo Typ\_des\_Charts - einer der Werte von Aufzählung [ENUM\\_CHART\\_MODE](#).

### ENUM\_CHART\_MODE

| Identifikator | Beschreibung                                                  |
|---------------|---------------------------------------------------------------|
| CHART_BARS    | Darstellung in Form von Bars                                  |
| CHART_CANDLES | Darstellung in Form von japanischen Kerzen                    |
| CHART_LINE    | Darstellung in Form einer Linie, gezogen an den Preisen Close |

für Andeutung des Image-Betrieves auf dem Preischart wird die Funktion [ChartSetInteger](#)(handle\_des\_Charts, [CHART\\_SHOW\\_VOLUMES](#), Chart\_Typ) verwendet, wo Chart\_Typ einer der Werte von Aufzählung [ENUM\\_CHART\\_VOLUME\\_MODE](#) ist.

### ENUM\_CHART\_VOLUME\_MODE

| Identifikator     | Beschreibung          |
|-------------------|-----------------------|
| CHART_VOLUME_HIDE | Volumen nicht gezeigt |
| CHART_VOLUME_TICK | Tickvolumen           |
| CHART_VOLUME_REAL | Handelsvolumen        |

### Beispiel:

```
//--- bekommen handle des laufenden Charts
long handle=ChartID();
if(handle>0) // wenn es gelungen ist, geben wir zuaetzliche Einstellungen vor
{
    //--- stellen wir auto-panning ab
    ChartSetInteger(handle,CHART_AUTOSCROLL,false);
    //--- tragen wir den Chart rechtsbündig ein
    ChartSetInteger(handle,CHART_SHIFT,true);
    //--- stellen wir in Form von Kerzen dar
    ChartSetInteger(handle,CHART_MODE,CHART_CANDLES);
    //--- laufen wir um 100 Bars vom Anfang der Geschichte
    //--- stellen wir Darstellung von Tickvolumen ein
    ChartSetInteger(handle,CHART_SHOW_VOLUMES,CHART_VOLUME_TICK);
};
```

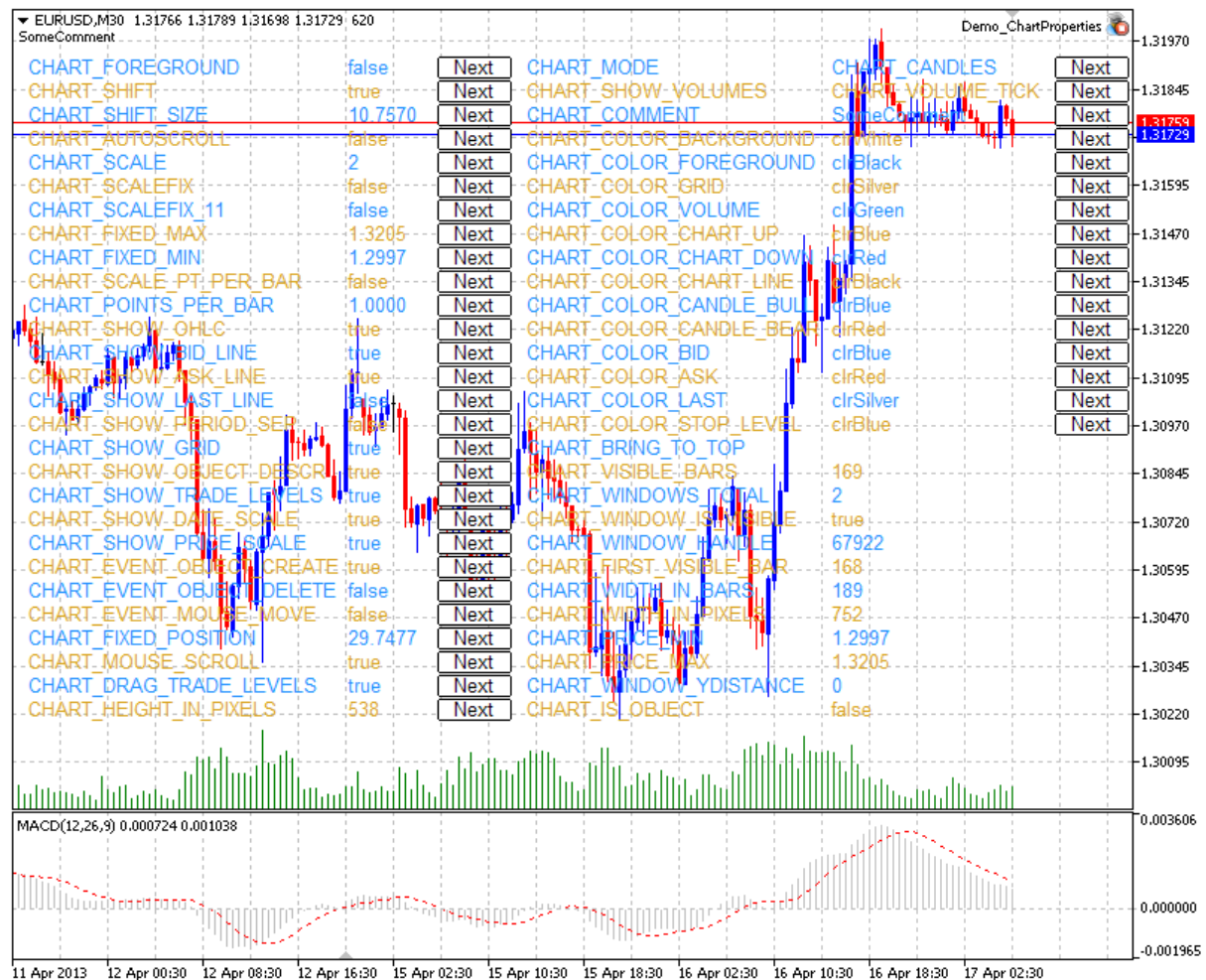
Sehen Sie auch

[ChartOpen](#), [ChartID](#)

## Beispiele von Arbeiten mit dem Chart

Dieser Abschnitt enthält Beispiele für die Eigenschaften des Charts. Für jede Eigenschaft gibt es hier ein oder zwei komplette Funktionen, die den Wert dieser Eigenschaft eingeben/erhalten erlauben. Diese Funktionen können in Ihrer MQL5 Programme verwendet werden.

Die Abbildung zeigt eine grafische Panel, die zeigt, wie Veränderung [des Charts](#) sein Aussehen ändert. Ein Klick auf "Next" erlaubt es Ihnen den neuen Wert der jeweiligen Eigenschaft einzugeben und die Änderungen im Chart sehen.



Quellcode der Panel ist [unten](#).

## Chart-Eigenschaften und Beispiele von Funktionen für Arbeit mit ihnen

- **CHART\_IS\_OBJECT** - bestimmt, ob das Objekt ein Chart oder [ein grafisches Objekt](#) ist.

```
//+-----+
//| Die Ermittlung, ob das Objekt ein Chart ist. Wenn es |
//| ein grafisches Objekt ist, ist das Ergebnis true. Wenn es ein |
//| Chart ist, wird das Variable result false gesetzt werden. |
//+-----+
```



```

bool ChartIsObject(bool &result,const long chart_ID=0)
{
//--- Bereiten wir eine Variable, um den Wert der Eigenschaft zu erhalten
    long value;
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Erhalten die Eigenschaft des Charts
    if(!ChartGetInteger(chart_ID,CHART_IS_OBJECT,0,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = "",GetLastError());
        //--- false zurückgeben
        return(false);
    }
//--- speichern den Wert der Chat-Eigenschaft in der Variable
    result=value;
//--- die erfolgreiche Umsetzung
    return(true);
};

```

- **CHART\_BRING\_TO\_TOP** - zeigt ein Chart über alle anderen.

```

//+-----+
//| Senden Befehl dem Terminal, Chart über alle anderen zu zeigen. |
//+-----+
bool ChartBringToTop(const long chart_ID=0)
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Zeigt wir den Chart über alle anderen
    if(!ChartSetInteger(chart_ID,CHART_BRING_TO_TOP,0,true))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = "",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
};

```

- **CHART\_MOUSE\_SCROLL** - Eigenschaft von Chart-Scroll mit der linken Maustaste.

```

//+-----+
//| Die Funktion ermittelt, ob Scroll des Charts mit der          |
//| linken Maustaste möglich ist.                                |
//+-----+
bool ChartMouseScrollGet(bool &result,const long chart_ID=0)

```

```

{
//--- Bereiten wir eine Variable, um den Wert der Eigenschaft zu erhalten
    long value;
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Erhalten wir den Wert der Eigenschaft
    if(!ChartGetInteger(chart_ID,CHART_MOUSE_SCROLL,0,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = ",GetLastError());
        return(false);
    }
//--- speichern den Wert der Chat-Eigenschaft in der Variable
    result=value;
//--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Die Funktion aktiviert/deaktiviert Scroll mit der linken Tasten |
//| möglich ist. |
//+-----+
bool ChartMouseScrollSet(const bool value,const long chart_ID=0)
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Geben wir den Wert der Eigenschaft ein
    if(!ChartSetInteger(chart_ID,CHART_MOUSE_SCROLL,0,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
};

```

- **CHART\_EVENT\_MOUSE\_MOVE** - Eigenschaft der Sendung von Nachrichten über Ereignisse der Bewegung und Klicken an den MQL5-Programmen ([CHARTEVENT\\_MOUSE\\_MOVE](#)).

```

//+-----+
//| Werden die Nachrichten über Ereignisse der Bewegung und Klicken |
//| an allen MQL5-Programmen auf diesem Chart gesendet? |
//+-----+
bool ChartEventMouseMoveGet(bool &result,const long chart_ID=0)
{
//--- Bereiten wir eine Variable, um den Wert der Eigenschaft zu erhalten
    long value;
//--- Setzen den Wert des Fehlers zurück

```

```

ResetLastError();
//--- Erhalten wir den Wert der Eigenschaft
if(!ChartGetInteger(chart_ID,CHART_EVENT_MOUSE_MOVE,0,value))
{
    //--- Schreiben die Fehlermeldung in den Log "Experten"
    Print(__FUNCTION__+"", Error Code = ",GetLastError());
    return(false);
};
//--- speichern den Wert der Chat-Eigenschaft in der Variable
result=value;
//--- die erfolgreiche Umsetzung
return(true);
};
//+-----+
//| Die Funktion aktiviert/deaktiviert das Senden von Nachrichten |
//| über Ereignisse von Bewegung und Klicken an allen |
//| MQL5-Programmen auf diesem Chart. |
//+-----+
bool ChartEventMouseMoveSet(const bool value,const long chart_ID=0)
{
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Geben wir den Wert der Eigenschaft ein
    if(!ChartSetInteger(chart_ID,CHART_EVENT_MOUSE_MOVE,0,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = ",GetLastError());
        return(false);
    };
    //--- die erfolgreiche Umsetzung
    return(true);
};

```

- **CHART\_EVENT\_OBJECT\_CREATE** - Eigenschaft der Sendung von Nachrichten über Ereignisse der Erstellung des grafischen Objektes an den MQL5-Programmen ([CHARTEVENT\\_OBJECT\\_CREATE](#)).

```

//+-----+
//| Werden die Nachrichten über Ereignisse der Erstellung eines |
//| Objektes an allen MQL5-Programmen auf diesem Chart gesendet? |
//+-----+
bool ChartEventObjectCreateGet(bool &result,const long chart_ID=0)
{
    //--- Bereiten wir eine Variable, um den Wert der Eigenschaft zu erhalten
    long value;
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Erhalten wir den Wert der Eigenschaft
    if(!ChartGetInteger(chart_ID,CHART_EVENT_OBJECT_CREATE,0,value))

```

```

    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = ", GetLastError());
        return(false);
    }
    //--- speichern den Wert der Chat-Eigenschaft in der Variable
    result=value;
    //--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Die Funktion aktiviert/deaktiviert das Senden von Nachrichten |
//| über Ereignisse von Erstellung eines grafischen Objektes an allen|
//| MQL5-Programmen auf diesem Chart.                             |
//+-----+
bool ChartEventObjectCreateSet(const bool value,const long chart_ID=0)
{
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Geben wir den Wert der Eigenschaft ein
    if(!ChartSetInteger(chart_ID,CHART_EVENT_OBJECT_CREATE,0,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = ", GetLastError());
        return(false);
    }
    //--- die erfolgreiche Umsetzung
    return(true);
};

```

- **CHART\_EVENT\_OBJECT\_DELETE** - Eigenschaft der Sendung von Nachrichten über Ereignisse der Löschung des grafischen Objektes an den MQL5-Programmen ([CHART\\_EVENT\\_OBJECT\\_DELETE](#)).

```

//+-----+
//| Werden die Nachrichten über Ereignisse der Löschung eines grafischen Objektes|
//| an allen MQL5-Programmen auf diesem Chart gesendet?                  |
//+-----+
bool ChartEventObjectDeleteGet(bool &result,const long chart_ID=0)
{
    //--- Bereiten wir eine Variable, um den Wert der Eigenschaft zu erhalten
    long value;
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Erhalten wir den Wert der Eigenschaft
    if(!ChartGetInteger(chart_ID,CHART_EVENT_OBJECT_DELETE,0,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = ", GetLastError());
    }
}

```

```

        return(false);
    };
//--- speichern den Wert der Chat-Eigenschaft in der Variable
    result=value;
//--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Die Funktion aktiviert/deaktiviert das Senden von Nachrichten über Ereignisse |
//| von Löschung eines grafischen Objektes an allen MQL5-Programmen auf diesem |
//| Chart.                               |
//+-----+
bool ChartEventObjectDeleteSet(const bool value,const long chart_ID=0)
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Geben wir den Wert der Eigenschaft ein
    if(!ChartSetInteger(chart_ID,CHART_EVENT_OBJECT_DELETE,0,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = "",GetLastError());
        return(false);
    };
//--- die erfolgreiche Umsetzung
    return(true);
};

```

- **CHART\_MODE** - Typ des Charts (Kerzen, Bars oder eine Linie).

```

//+-----+
//| Bekommen wir den Typ des Charts (in Form von Kerzen, Bars oder |
//| einer Linie).                               |
//+-----+
ENUM_CHART_MODE ChartModeGet(const long chart_ID=0)
{
//--- Bereiten wir eine Variable, um den Wert der Eigenschaft zu erhalten
    long result=WRONG_VALUE;
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Erhalten wir den Wert der Eigenschaft
    if(!ChartGetInteger(chart_ID,CHART_MODE,0,result))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = "",GetLastError());
    };
//--- Geben den Wert der Eigenschaft zurück
    return((ENUM_CHART_MODE)result);
};

```

```
//+-----+
//| Setzen den Typ des Charts (in Form von Kerzen, Bars oder |
//| einer Linie).                                         |
//+-----+
bool ChartModeSet(const long value,const long chart_ID=0)
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Geben wir den Wert der Eigenschaft ein
    if(!ChartSetInteger(chart_ID,CHART_MODE,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
};
```

- **CHART\_FOREGROUND** - Eigenschaft der Anzeige vom Preischart im Vordergrund.

```
//+-----+
//| Die Funktion ermittelt, ob das Preischart im Vordergrund angezeigt |
//| wird.                                                         |
//+-----+
bool ChartForegroundGet(bool &result,const long chart_ID=0)
{
//--- Bereiten wir eine Variable, um den Wert der Eigenschaft zu erhalten
    long value;
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Erhalten wir den Wert der Eigenschaft
    if(!ChartGetInteger(chart_ID,CHART_FOREGROUND,0,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
        return(false);
    }
//--- speichern den Wert der Chat-Eigenschaft in der Variable
    result=value;
//--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Die Funktion aktiviert/deaktiviert Anzeige des Preischarts im |
//| Vordergrund.                                                         |
//+-----+
bool ChartForegroundSet(const bool value,const long chart_ID=0)
```

```

{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Geben wir den Wert der Eigenschaft ein
    if(!ChartSetInteger(chart_ID,CHART_FOREGROUND,0,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
}

```

- **CHART\_SHIFT** - Modus von der Verschiebung des Preischarts vom rechten Rand.

```

//+-----+
//| Die Funktion ermittelt, ob die Modus der Anzeige des Charts mit |
//| Verschiebung vom rechten Rand aktiv ist. |
//+-----+
bool ChartShiftGet(bool &result,const long chart_ID=0)
{
//--- Bereiten wir eine Variable, um den Wert der Eigenschaft zu erhalten
    long value;
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Erhalten wir den Wert der Eigenschaft
    if(!ChartGetInteger(chart_ID,CHART_SHIFT,0,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
        return(false);
    }
//--- speichern den Wert der Chat-Eigenschaft in der Variable
    result=value;
//--- die erfolgreiche Umsetzung
    return(true);
}

//+-----+
//| Die Funktion aktiviert/deaktiviert Anzeige des Preischarts mit |
//| Verschiebung vom rechten Rand. |
//+-----+
bool ChartShiftSet(const bool value,const long chart_ID=0)
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Geben wir den Wert der Eigenschaft ein
    if(!ChartSetInteger(chart_ID,CHART_SHIFT,0,value))

```

```

    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = ",GetLastError());
        return(false);
    };
//--- die erfolgreiche Umsetzung
    return(true);
; }

```

- **CHART\_AUTOSCROLL** - Modus der automatischen Scrollen an den rechten Rand des Charts.

```

//+-----+
//| Die Funktion ermittelt, ob die Modus der automatischen Scrollen des Charts |
//| nach rechts, wenn neue Ticks empfangen werden, aktiv ist.                  |
//+-----+
bool ChartAutoscrollGet(bool &result,const long chart_ID=0)
{
    //--- Bereiten wir eine Variable, um den Wert der Eigenschaft zu erhalten
    long value;
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Erhalten wir den Wert der Eigenschaft
    if(!ChartGetInteger(chart_ID,CHART_AUTOSCROLL,0,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = ",GetLastError());
        return(false);
    };
    //--- speichern den Wert der Chat-Eigenschaft in der Variable
    result=value;
    //--- die erfolgreiche Umsetzung
    return(true);
; }

//+-----+
//| Die Funktion aktiviert/deaktiviert die automatische Scrollen des Charts |
//| nach rechts, wenn neue Ticks empfangen werden.                          |
//+-----+
bool ChartAutoscrollSet(const bool value,const long chart_ID=0)
{
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Geben wir den Wert der Eigenschaft ein
    if(!ChartSetInteger(chart_ID,CHART_AUTOSCROLL,0,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = ",GetLastError());
        return(false);
    };
; }

```



```
//--- die erfolgreiche Umsetzung
    return(true);
;}
```

- **CHART\_SCALE** - Eigenschaft der Chart-Skala.

```
//+-----+
//| Bekommen die Skala des Charts (0 bis 5). |
//+-----+
int ChartScaleGet(const long chart_ID=0)
{
//--- Bereiten wir eine Variable, um den Wert der Eigenschaft zu erhalten
    long result=-1;
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Erhalten wir den Wert der Eigenschaft
    if(!ChartGetInteger(chart_ID,CHART_SCALE,0,result))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
    }
//--- Geben den Wert der Eigenschaft zurück
    return((int)result);
;}
```

```
//+-----+
//| Setzen die Skala des Charts (0 bis 5). |
//+-----+
bool ChartScaleSet(const long value,const long chart_ID=0)
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Geben wir den Wert der Eigenschaft ein
    if(!ChartSetInteger(chart_ID,CHART_SCALE,0,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
;}
```

- **CHART\_SCALEFIX** - Modus der festen Skala des Charts.

```
//+-----+
//| Die Funktion ermittelt, ob die Modus der festen Skala aktiv ist. |
//+-----+
```

```

bool ChartScaleFixGet(bool &result,const long chart_ID=0)
{
    //--- Bereiten wir eine Variable, um den Wert der Eigenschaft zu erhalten
    long value;
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Erhalten wir den Wert der Eigenschaft
    if(!ChartGetInteger(chart_ID,CHART_SCALEFIX,0,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = ",GetLastError());
        return(false);
    }
    //--- speichern den Wert der Chat-Eigenschaft in der Variable
    result=value;
    //--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Die Funktion aktiviert/deaktiviert die feste Skala |
//+-----+
bool ChartScaleFixSet(const bool value,const long chart_ID=0)
{
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Geben wir den Wert der Eigenschaft ein
    if(!ChartSetInteger(chart_ID,CHART_SCALEFIX,0,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = ",GetLastError());
        return(false);
    }
    //--- die erfolgreiche Umsetzung
    return(true);
};

```

- CHART\_SCALEFIX\_11 - Modus der Chartskala 1:1.

```

//+-----+
//| Die Funktion ermittelt, ob die Modus der Skala "1:1" aktiv ist. |
//+-----+
bool ChartScaleFix11Get(bool &result,const long chart_ID=0)
{
    //--- Bereiten wir eine Variable, um den Wert der Eigenschaft zu erhalten
    long value;
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Erhalten wir den Wert der Eigenschaft

```

```

    if(!ChartGetInteger(chart_ID,CHART_SCALEFIX_11,0,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = ",GetLastError());
        return(false);
    }
    ;}
//--- speichern den Wert der Chat-Eigenschaft in der Variable
    result=value;
//--- die erfolgreiche Umsetzung
    return(true);
    ;}
//+-----+
//| Die Funktion aktiviert/deaktiviert die Skala "1:1" |
//+-----+
bool ChartScaleFix11Set(const bool value,const long chart_ID=0)
{
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Geben wir den Wert der Eigenschaft ein
    if(!ChartSetInteger(chart_ID,CHART_SCALEFIX_11,0,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = ",GetLastError());
        return(false);
    }
    ;}
//--- die erfolgreiche Umsetzung
    return(true);
    ;}

```

- **CHART\_SCALE\_PT\_PER\_BAR** - Modus der Eingabe der Skala in Punkten pro Bar.

```

//+-----+
//| Die Funktion ermittelt, ob die Modus der Eingabe der Skala in Punkten |
//| pro Bar aktiv ist. |
//+-----+
bool ChartScalePerBarGet(bool &result,const long chart_ID=0)
{
    //--- Bereiten wir eine Variable, um den Wert der Eigenschaft zu erhalten
    long value;
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Erhalten wir den Wert der Eigenschaft
    if(!ChartGetInteger(chart_ID,CHART_SCALE_PT_PER_BAR,0,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = ",GetLastError());
        return(false);
    }
    ;}

```

```

//--- speichern den Wert der Chat-Eigenschaft in der Variable
    result=value;
//--- die erfolgreiche Umsetzung
    return(true);
; }

//+-----+
//| Die Funktion aktiviert/deaktiviert Modus der Eingabe der Skala in Punkten |
//| pro Bar.                                                                    |
//+-----+
bool ChartScalePerBarSet(const bool value,const long chart_ID=0)
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Geben wir den Wert der Eigenschaft ein
    if(!ChartSetInteger(chart_ID,CHART_SCALE_PT_PER_BAR,0,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
; }

```

- **CHART\_SHOW\_OHLC** - Eigenschaft der Anzeige der Werten OHLC in der linken oberen Ecke.

```

//+-----+
//| Die Funktion ermittelt, ob die Modus der Anzeige der Werten OHLC in der linken|
//| oberen Ecke aktiv ist.                                                        |
//+-----+
bool ChartShowOHLCGet(bool &result,const long chart_ID=0)
{
//--- Bereiten wir eine Variable, um den Wert der Eigenschaft zu erhalten
    long value;
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Erhalten wir den Wert der Eigenschaft
    if(!ChartGetInteger(chart_ID,CHART_SHOW_OHLC,0,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
        return(false);
    }
//--- speichern den Wert der Chat-Eigenschaft in der Variable
    result=value;
//--- die erfolgreiche Umsetzung
    return(true);
; }

```

```
//+-----+
//| Die Funktion aktiviert/deaktiviert Anzeige der Werten OHLC in der |
//| linken oberen Ecke des Charts.                                     |
//+-----+
bool ChartShowOHLCSet(const bool value,const long chart_ID=0)
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Geben wir den Wert der Eigenschaft ein
    if(!ChartSetInteger(chart_ID,CHART_SHOW_OHLC,0,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
};
```

- **CHART\_SHOW\_BID\_LINE** - Eigenschaft der Anzeige des Bid-Wertes als eine horizontale Linie auf dem Chart.

```
//+-----+
//| Die Funktion ermittelt die Modus der Anzeige der Bid-Linie auf dem |
//| Chart.                                                             |
//+-----+
bool ChartShowBidLineGet(bool &result,const long chart_ID=0)
{
//--- Bereiten wir eine Variable, um den Wert der Eigenschaft zu erhalten
    long value;
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Erhalten wir den Wert der Eigenschaft
    if(!ChartGetInteger(chart_ID,CHART_SHOW_BID_LINE,0,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
        return(false);
    }
//--- speichern den Wert der Chat-Eigenschaft in der Variable
    result=value;
//--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Die Funktion aktiviert/deaktiviert Anzeige der Bid-Linie auf dem |
//| Chart.                                                             |
//+-----+
```

```

bool ChartShowBidLineSet(const bool value,const long chart_ID=0)
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Geben wir den Wert der Eigenschaft ein
    if(!ChartSetInteger(chart_ID,CHART_SHOW_BID_LINE,0,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
};

```

- **CHART\_SHOW\_ASK\_LINE** - Eigenschaft der Anzeige des Ask-Wertes als eine horizontale Linie auf dem Chart.

```

//+-----+
//| Die Funktion ermittelt die Modus der Anzeige der Ask-Linie auf dem |
//| Chart. |
//+-----+
bool ChartShowAskLineGet(bool &result,const long chart_ID=0)
{
//--- Bereiten wir eine Variable, um den Wert der Eigenschaft zu erhalten
    long value;
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Erhalten wir den Wert der Eigenschaft
    if(!ChartGetInteger(chart_ID,CHART_SHOW_ASK_LINE,0,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
        return(false);
    }
//--- speichern den Wert der Chat-Eigenschaft in der Variable
    result=value;
//--- die erfolgreiche Umsetzung
    return(true);
};

//+-----+
//| Die Funktion aktiviert/deaktiviert Anzeige der Ask-Linie auf dem |
//| Chart. |
//+-----+
bool ChartShowAskLineSet(const bool value,const long chart_ID=0)
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();

```

```
//--- Geben wir den Wert der Eigenschaft ein
if(!ChartSetInteger(chart_ID,CHART_SHOW_ASK_LINE,0,value))
{
    //--- Schreiben die Fehlermeldung in den Log "Experten"
    Print(__FUNCTION__+"", Error Code = "",GetLastError());
    return(false);
};
//--- die erfolgreiche Umsetzung
return(true);
};
```

- **CHART\_SHOW\_LAST\_LINE** - Eigenschaft der Anzeige des Last-Wertes als eine horizontale Linie auf dem Chart.

```
//+-----+
//| Die Funktion ermittelt, ob die Modus der Anzeige der Linie für den Preis der letzten |
//| Transaktion aktiv ist. |
//+-----+
bool ChartShowLastLineGet(bool &result,const long chart_ID=0)
{
    //--- Bereiten wir eine Variable, um den Wert der Eigenschaft zu erhalten
    long value;
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Erhalten wir den Wert der Eigenschaft
    if(!ChartGetInteger(chart_ID,CHART_SHOW_LAST_LINE,0,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = "",GetLastError());
        return(false);
    };
    //--- speichern den Wert der Chat-Eigenschaft in der Variable
    result=value;
    //--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Die Funktion aktiviert/deaktiviert Anzeige der Linie des Preises der letzten |
//| Transaktion. |
//+-----+
bool ChartShowLastLineSet(const bool value,const long chart_ID=0)
{
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Geben wir den Wert der Eigenschaft ein
    if(!ChartSetInteger(chart_ID,CHART_SHOW_LAST_LINE,0,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
```

```

        Print(__FUNCTION__+"", Error Code = "",GetLastError());
        return(false);
    };
//--- die erfolgreiche Umsetzung
    return(true);
;}
```

- **CHART\_SHOW\_PERIOD\_SEP** - Eigenschaft der Anzeige der vertikalen Teilern zwischen benachbarten Perioden.

```

//+-----+
//| Die Funktion ermittelt die Modus der Anzeige der vertikalen Teilern |
//| zwischen benachbarten Perioden.                                     |
//+-----+
bool ChartShowPeriodSeparatorGet(bool &result,const long chart_ID=0)
{
//--- Bereiten wir eine Variable, um den Wert der Eigenschaft zu erhalten
    long value;
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Erhalten wir den Wert der Eigenschaft
    if(!ChartGetInteger(chart_ID,CHART_SHOW_PERIOD_SEP,0,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = "",GetLastError());
        return(false);
    };
//--- speichern den Wert der Chat-Eigenschaft in der Variable
    result=value;
//--- die erfolgreiche Umsetzung
    return(true);
;}
```

```

//+-----+
//| Die Funktion aktiviert/deaktiviert die Anzeige der vertikalen Teilern |
//| zwischen benachbarten Perioden.                                     |
//+-----+
bool ChartShowPeriodSepapatorSet(const bool value,const long chart_ID=0)
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Geben wir den Wert der Eigenschaft ein
    if(!ChartSetInteger(chart_ID,CHART_SHOW_PERIOD_SEP,0,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = "",GetLastError());
        return(false);
    };
//--- die erfolgreiche Umsetzung
```



```

    return(true);
; }

```

- **CHART\_SHOW\_GRID** - Eigenschaft der Anzeige des Chartrasters.

```

//+-----+
//| Die Funktion ermittelt, ob der Chartraster angezeigt wird. |
//+-----+
bool ChartShowGridGet(bool &result,const long chart_ID=0)
{
//--- Bereiten wir eine Variable, um den Wert der Eigenschaft zu erhalten
    long value;
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Erhalten wir den Wert der Eigenschaft
    if(!ChartGetInteger(chart_ID,CHART_SHOW_GRID,0,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
        return(false);
    }
//--- speichern den Wert der Chat-Eigenschaft in der Variable
    result=value;
//--- die erfolgreiche Umsetzung
    return(true);
; }

//+-----+
//| Die Funktion aktiviert/deaktiviert den Chartraster. |
//+-----+
bool ChartShowGridSet(const bool value,const long chart_ID=0)
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Geben wir den Wert der Eigenschaft ein
    if(!ChartSetInteger(chart_ID,CHART_SHOW_GRID,0,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
; }

```

- **CHART\_SHOW\_VOLUMES** - Eigenschaft der Anzeige der Volumen auf dem Chart.

```

//+-----+

```

```

//| Die Funktion ermittelt, ob die Volumen auf dem Chart angezeigt werden (werden |
//| nicht angezeigt, Tick-Volumen angezeigt werden, reale Volumen angezeigt werden).
//+-----+
ENUM_CHART_VOLUME_MODE ChartShowVolumesGet(const long chart_ID=0)
{
//--- Bereiten wir eine Variable, um den Wert der Eigenschaft zu erhalten
    long result=WRONG_VALUE;
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Erhalten wir den Wert der Eigenschaft
    if(!ChartGetInteger(chart_ID,CHART_SHOW_VOLUMES,0,result))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
    }
//--- Geben den Wert der Eigenschaft zurück
    return((ENUM_CHART_VOLUME_MODE)result);
;}

//+-----+
//| Die Funktion setzt den Modus der Anzeige der Volumina auf dem Chart.
//+-----+
bool ChartShowVolumesSet(const long value,const long chart_ID=0)
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Geben wir den Wert der Eigenschaft ein
    if(!ChartSetInteger(chart_ID,CHART_SHOW_VOLUMES,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
;}

```

- **CHART\_SHOW\_OBJECT\_DESCR** - Eigenschaft der Pop-up-Beschreibungen von grafischen Objekten.

```

//+-----+
//| Die Funktion ermittelt, ob die Pop-up-Beschreibungen der grafischen Objekten angezei
//| wenn Sie mit dem Mauszeiger auf ihnen deuten.
//+-----+
bool ChartShowObjectDescriptionGet(bool &result,const long chart_ID=0)
{
//--- Bereiten wir eine Variable, um den Wert der Eigenschaft zu erhalten
    long value;
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();

```

```

//--- Erhalten wir den Wert der Eigenschaft
if(!ChartGetInteger(chart_ID,CHART_SHOW_OBJECT_DESCR,0,value))
{
    //--- Schreiben die Fehlermeldung in den Log "Experten"
    Print(__FUNCTION__+" Error Code = ",GetLastError());
    return(false);
};
//--- speichern den Wert der Chat-Eigenschaft in der Variable
result=value;
//--- die erfolgreiche Umsetzung
return(true);
};
//+-----+
//| Die Funktion aktiviert/deaktiviert die Anzeige der Popup-Beschreibungen der grafis
//| wenn Sie mit dem Mauszeiger auf ihnen deuten. |
//+-----+
bool ChartShowObjectDescriptionSet(const bool value,const long chart_ID=0)
{
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Geben wir den Wert der Eigenschaft ein
    if(!ChartSetInteger(chart_ID,CHART_SHOW_OBJECT_DESCR,0,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
        return(false);
    };
    //--- die erfolgreiche Umsetzung
    return(true);
};

```

- **CHART\_VISIBLE\_BARS** - ermittelt die Anzahl der Baren auf dem Chart, die zur Anzeige verfügbar sind.

```

//+-----+
//| Die Funktion bekommt die Anzahl der Bars, die angezeigt (sichtbar) wird |
//| im Chart-Fenster. |
//+-----+
int ChartVisibleBars(const long chart_ID=0)
{
    //--- Bereiten wir eine Variable, um den Wert der Eigenschaft zu erhalten
    long result=-1;
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Erhalten wir den Wert der Eigenschaft
    if(!ChartGetInteger(chart_ID,CHART_VISIBLE_BARS,0,result))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
    }
}

```

```

        Print(__FUNCTION__+"", Error Code = ",GetLastError());
    ;}

//--- Geben den Wert der Eigenschaft zurück
    return((int)result);
;}
```

- **CHART\_WINDOWS\_TOTAL** - ermittelt die Gesamtzahl der Chart-Fenster, einschließlich Unterfenster der Indikatoren.

```

//+-----+
//| Die Funktion erhält die Gesamtzahl der Chart-Fenster, einschließlich Unterfenster |
//| der Indikatoren. |
//+-----+
int ChartWindowsTotal(const long chart_ID=0)
{
//--- Bereiten wir eine Variable, um den Wert der Eigenschaft zu erhalten
    long result=-1;
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Erhalten wir den Wert der Eigenschaft
    if(!ChartGetInteger(chart_ID,CHART_WINDOWS_TOTAL,0,result))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = ",GetLastError());
    ;}
//--- Geben den Wert der Eigenschaft zurück
    return((int)result);
;}
```

- **CHART\_WINDOW\_IS\_VISIBLE** - ermittelt die Sichtbarkeit des Unterfensters.

```

//+-----+
//| Die Funktion ermittelt, ob ein Fenster oder Unterfenster des Charts |
//| sichtbar ist. |
//+-----+
bool ChartWindowsIsVisible(bool &result,const long chart_ID=0,const int sub_window=0)
{
//--- Bereiten wir eine Variable, um den Wert der Eigenschaft zu erhalten
    long value;
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Erhalten wir den Wert der Eigenschaft
    if(!ChartGetInteger(chart_ID,CHART_WINDOW_IS_VISIBLE,sub_window,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = ",GetLastError());
        return(false);
    }
```

```

    ;}
//--- speichern den Wert der Chat-Eigenschaft in der Variable
    result=value;
//--- die erfolgreiche Umsetzung
    return(true);
    ;}

```

- **CHART\_WINDOW\_HANDLE** - gibt ein Handle des Charts zurück.

```

//+-----+
//| Die Funktion erhält den Handle des Charts                                     |
//+-----+
int ChartWindowsHandle(const long chart_ID=0)
{
//--- Bereiten wir eine Variable, um den Wert der Eigenschaft zu erhalten
    long result=-1;
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Erhalten wir den Wert der Eigenschaft
    if(!ChartGetInteger(chart_ID,CHART_WINDOW_HANDLE,0,result))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
    }
//--- Geben den Wert der Eigenschaft zurück
    return((int)result);
    ;}

```

- **CHART\_WINDOW\_YDISTANCE** - ermittelt den Abstand in Pixel zwischen dem oberen Rahmen des Indikator-Unterfensters und dem oberen Rahmen des Hauptfensters des Charts.

```

//+-----+
//| Die Funktion erhält den Abstand in Pixel zwischen dem oberen Rahmen des      |
//| Unterfensters und oberen Rahmen des Hauptfensters.                            |
//+-----+
int ChartWindowsYDistance(const long chart_ID=0,const int sub_window=0)
{
//--- Bereiten wir eine Variable, um den Wert der Eigenschaft zu erhalten
    long result=-1;
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Erhalten wir den Wert der Eigenschaft
    if(!ChartGetInteger(chart_ID,CHART_WINDOW_YDISTANCE,sub_window,result))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
    }
    ;}

```

```
//--- Geben den Wert der Eigenschaft zurück
    return((int)result);
;}
```

- **CHART\_FIRST\_VISIBLE\_BAR** - gibt den Nummer des ersten sichtbaren Bars auf dem Chart zurück (Indizierung der Baren wie in [Zeitreihe](#)).

```
//+-----+
//| Die Funktion bekommt die Nummer des ersten sichtbaren Bars des Charts. |
//| Indexing als in Zeitreihe, dass die letzten Bars kleineren Indizes haben. |
//+-----+
int ChartFirstVisibleBar(const long chart_ID=0)
{
    //--- Bereiten wir eine Variable, um den Wert der Eigenschaft zu erhalten
    long result=-1;
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Erhalten wir den Wert der Eigenschaft
    if(!ChartGetInteger(chart_ID,CHART_FIRST_VISIBLE_BAR,0,result))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = "",GetLastError());
    ;}
    //--- Geben den Wert der Eigenschaft zurück
    return((int)result);
;}
```

- **CHART\_WIDTH\_IN\_BARS** - gibt die Breite des Chart in den Baren zurück.

```
//+-----+
//| Die Funktion erhält den Wert der Breite des Charts in den Baren. |
//+-----+
int ChartWidthInBars(const long chart_ID=0)
{
    //--- Bereiten wir eine Variable, um den Wert der Eigenschaft zu erhalten
    long result=-1;
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Erhalten wir den Wert der Eigenschaft
    if(!ChartGetInteger(chart_ID,CHART_WIDTH_IN_BARS,0,result))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = "",GetLastError());
    ;}
    //--- Geben den Wert der Eigenschaft zurück
    return((int)result);
;}
```

- **CHART\_WIDTH\_IN\_PIXELS** - gibt die Breite des Chart in den Pixeln zurück.

```
//+-----+
//| Die Funktion erhält den Wert der Breite des Charts in Pixeln. |
//+-----+
int ChartWidthInPixels(const long chart_ID=0)
{
    //-- Bereiten wir eine Variable, um den Wert der Eigenschaft zu erhalten
    long result=-1;
    //-- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //-- Erhalten wir den Wert der Eigenschaft
    if(!ChartGetInteger(chart_ID,CHART_WIDTH_IN_PIXELS,0,result))
    {
        //-- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
    }
    //-- Geben den Wert der Eigenschaft zurück
    return((int)result);
}
```

- **CHART\_HEIGHT\_IN\_PIXELS** - Eigenschaft der Höhe des Charts in Pixeln.

```
//+-----+
//| Die Funktion erhält den Wert der Höhe des Charts in Pixeln. |
//+-----+
int ChartHeightInPixelsGet(const long chart_ID=0,const int sub_window=0)
{
    //-- Bereiten wir eine Variable, um den Wert der Eigenschaft zu erhalten
    long result=-1;
    //-- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //-- Erhalten wir den Wert der Eigenschaft
    if(!ChartGetInteger(chart_ID,CHART_HEIGHT_IN_PIXELS,sub_window,result))
    {
        //-- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
    }
    //-- Geben den Wert der Eigenschaft zurück
    return((int)result);
}

//+-----+
//| Die Funktion setzt den Wert der Höhe des Charts in Pixeln. |
//+-----+
bool ChartHeightInPixelsSet(const int value,const long chart_ID=0,const int sub_window=0)
{
    // ...
}
```

```
//--- Setzen den Wert des Fehlers zurück
ResetLastError();
//--- Geben wir den Wert der Eigenschaft ein
if(!ChartSetInteger(chart_ID, CHART_HEIGHT_IN_PIXELS, sub_window, value))
{
    //--- Schreiben die Fehlermeldung in den Log "Experten"
    Print(__FUNCTION__+"", Error Code = "", GetLastError());
    return(false);
};
//--- die erfolgreiche Umsetzung
return(true);
};
```

- **CHART\_COLOR\_BACKGROUND** - Farbe von Hintergrund des Charts.

```
//+-----+
//| Die Funktion erhält die Hintergrundfarbe des Charts.
//+-----+
color ChartBackColorGet(const long chart_ID=0)
{
    //--- Bereiten wir eine Variable, um die Farbe zu erhalten
    long result=clrNONE;
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Erhalten wir die Hintergrundfarbe des Charts
    if(!ChartGetInteger(chart_ID, CHART_COLOR_BACKGROUND, 0, result))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = "", GetLastError());
    };
    //--- Geben den Wert der Eigenschaft zurück
    return((color)result);
};
//+-----+
//| Die Funktion setzt die Hintergrundfarbe des Charts.
//+-----+
bool ChartBackColorSet(const color clr, const long chart_ID=0)
{
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Setzen wir die Hintergrundfarbe des Charts
    if(!ChartSetInteger(chart_ID, CHART_COLOR_BACKGROUND, clr))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = "", GetLastError());
        return(false);
    };
    //--- die erfolgreiche Umsetzung
```



```

    return(true);
; }

```

- **CHART\_COLOR\_FOREGROUND** - Farbe der Achse, Skala und OHLC Zeile.

```

//+-----+
//| Die Funktion erhält die Farbe der Achse, Skala und OHLC Zeile. |
//+-----+
color ChartForeColorGet(const long chart_ID=0)
{
//--- Bereiten wir eine Variable, um die Farbe zu erhalten
    long result=clrNONE;
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Erhalten wir die Farbe der Achse, Skala und OHLC Zeile
    if(!ChartGetInteger(chart_ID,CHART_COLOR_FOREGROUND,0,result))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
    ;}
//--- Geben den Wert der Eigenschaft zurück
    return((color)result);
; }

//+-----+
//| Die Funktion setzt die Farbe der Achse, Skala und OHLC Zeile. |
//+-----+
bool ChartForeColorSet(const color clr,const long chart_ID=0)
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Setzen wir die Farbe der Achse, Skala und OHLC Zeile
    if(!ChartSetInteger(chart_ID,CHART_COLOR_FOREGROUND,clr))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
        return(false);
    ;}
//--- die erfolgreiche Umsetzung
    return(true);
; }

```

- **CHART\_COLOR\_GRID** - Farbe des Chart-Gitters.

```

//+-----+
//| Die Funktion erhält die Farbe des Gitters des Charts. |
//+-----+
color ChartGridColorGet(const long chart_ID=0)

```

```

{
//--- Bereiten wir eine Variable, um die Farbe zu erhalten
    long result=clrNONE;
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Erhalten die Farbe des Chart-Gitters
    if(!ChartGetInteger(chart_ID,CHART_COLOR_GRID,0,result))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
    }
//--- Geben den Wert der Eigenschaft zurück
    return((color)result);
;}

//+-----+
//| Die Funktion setzt die Farbe des Gitters des Charts. |
//+-----+
bool ChartGridColorSet(const color clr,const long chart_ID=0)
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Setzen wir die Farbe des Chart-Gitters
    if(!ChartSetInteger(chart_ID,CHART_COLOR_GRID,clr))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
;}

```

- **CHART\_COLOR\_VOLUME** - Farbe der Volumen und Ebenen der Öffnung der Positionen.

```

//+-----+
//| Die Funktion bekommt die Farbe Volumen oder Ebenen der Öffnung |
//| der Positionen. |
//+-----+
color ChartVolumeColorGet(const long chart_ID=0)
{
//--- Bereiten wir eine Variable, um die Farbe zu erhalten
    long result=clrNONE;
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Erhalten wir die Farbe der Volumen und Ebenen der Öffnung der Positionen
    if(!ChartGetInteger(chart_ID,CHART_COLOR_VOLUME,0,result))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"

```

```

        Print(__FUNCTION__+"", Error Code = "",GetLastError());
    ;}

//--- Geben den Wert der Eigenschaft zurück
    return((color)result);
; }

//+-----+
//| Die Funktion setzt die Farbe Volumen oder Ebenen der Öffnung |
//| der Positionen.                                             |
//+-----+
bool ChartVolumeColorSet(const color clr,const long chart_ID=0)
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Setzen wir die Farbe der Volumen und Ebenen der Öffnung der Positionen
    if(!ChartSetInteger(chart_ID,CHART_COLOR_VOLUME,clr))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = "",GetLastError());
        return(false);
    ;}
//--- die erfolgreiche Umsetzung
    return(true);
; }

```

- **CHART\_COLOR\_CHART\_UP** - Farbe des Bars nach oben, Schatten und Kanten des Körpers von Bull-Kerze.

```

//+-----+
//| Die Funktion bekommt die Farbe des Bars nach oben, die Farbe der Schatten und |
//| Kanten des Körpers von Bull-Kerze.                                             |
//+-----+
color ChartUpColorGet(const long chart_ID=0)
{
//--- Bereiten wir eine Variable, um die Farbe zu erhalten
    long result=clrNONE;
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- erhalten die Farbe des Bars nach oben, Schatten und Kanten des Körpers von Bull-
    if(!ChartGetInteger(chart_ID,CHART_COLOR_CHART_UP,0,result))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = "",GetLastError());
    ;}
//--- Geben den Wert der Eigenschaft zurück
    return((color)result);
; }

//+-----+
//| Die Funktion setzt die Farbe des Bars nach oben, die Farbe der Schatten und |

```

```

//| Kanten des Körpers von Bull-Kerze. |
//+-----+
bool ChartUpColorSet(const color clr,const long chart_ID=0)
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- setzen die Farbe des Bars nach oben, Schatten und Kanten des Körpers von Bull-Kerze
    if(!ChartSetInteger(chart_ID,CHART_COLOR_CHART_UP,clr))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
}

```

- **CHART\_COLOR\_CHART\_DOWN** - Farbe des Bars nach unten, Schatten und Kanten des Körpers von Bär-Kerze.

```

//+-----+
//| Die Funktion bekommt die Farbe des Bars nach unten, die Farbe der Schatten und |
//| Kanten des Körpers von Bär-Kerze. |
//+-----+
color ChartDownColorGet(const long chart_ID=0)
{
//--- Bereiten wir eine Variable, um die Farbe zu erhalten
    long result=clrNONE;
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- erhalten die Farbe des Bars nach unten, Schatten und Kanten des Körpers von Bär-Kerze
    if(!ChartGetInteger(chart_ID,CHART_COLOR_CHART_DOWN,0,result))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = ",GetLastError());
    }
//--- Geben den Wert der Eigenschaft zurück
    return((color)result);
}
//+-----+
//| Die Funktion setzt die Farbe des Bars nach unten, die Farbe der Schatten und |
//| Kanten des Körpers von Bär-Kerze. |
//+-----+
bool ChartDownColorSet(const color clr,const long chart_ID=0)
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- setzen die Farbe des Bars nach unten, Schatten und Kanten des Körpers von Bär-Kerze

```

```

    if(!ChartSetInteger(chart_ID,CHART_COLOR_CHART_DOWN,clr))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
};

```

- **CHART\_COLOR\_CHART\_LINE** - Farbe der Linie des Charts und Kerzen "Doji".

```

//+-----+
//| Die Funktion erhält die Farbe der Linie des Charts und Kerzen "Doji". |
//+-----+
color ChartLineColorGet(const long chart_ID=0)
{
    //--- Bereiten wir eine Variable, um die Farbe zu erhalten
    long result=clrNONE;
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Erhalten wir die Farbe der Linie des Charts und Kerzen "Doji"
    if(!ChartGetInteger(chart_ID,CHART_COLOR_CHART_LINE,0,result))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = ",GetLastError());
    }
    //--- Geben den Wert der Eigenschaft zurück
    return((color)result);
};
//+-----+
//| Die Funktion setzt die Farbe der Linie des Charts und Kerzen |
//| "Doji". |
//+-----+
bool ChartLineColorSet(const color clr,const long chart_ID=0)
{
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Setzen wir die Farbe der Linie des Charts und Kerzen "Doji"
    if(!ChartSetInteger(chart_ID,CHART_COLOR_CHART_LINE,clr))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = ",GetLastError());
        return(false);
    }
    //--- die erfolgreiche Umsetzung
    return(true);
};

```

- **CHART\_COLOR\_CANDLE\_BULL** - Farbe des Körpers von Bull-Kerze.

```
//+-----+
//| Die Funktion erhält die Farbe des Körpers von Bull-Kerze.
//+-----+
color ChartBullColorGet(const long chart_ID=0)
{
//--- Bereiten wir eine Variable, um die Farbe zu erhalten
    long result=clrNONE;
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- erhalten die Farbe des Körpers von Bull-Kerze
    if(!ChartGetInteger(chart_ID,CHART_COLOR_CANDLE_BULL,0,result))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
    }
//--- Geben den Wert der Eigenschaft zurück
    return((color)result);
;}

//+-----+
//| Die Funktion setzt die Farbe des Körpers von Bull-Kerze.
//+-----+
bool ChartBullColorSet(const color clr,const long chart_ID=0)
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- setzen die Farbe des Körpers von Bull-Kerze
    if(!ChartSetInteger(chart_ID,CHART_COLOR_CANDLE_BULL,clr))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
;}
```

- **CHART\_COLOR\_CANDLE\_BEAR** - Farbe des Körpers von Bär-Kerze.

```
//+-----+
//| Die Funktion erhält die Farbe des Körpers von Bär-Kerze.
//+-----+
color ChartBearColorGet(const long chart_ID=0)
{
//--- Bereiten wir eine Variable, um die Farbe zu erhalten
```

```

    long result=clrNONE;
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- erhalten die Farbe des Körpers von Bär-Kerze
    if(!ChartGetInteger(chart_ID,CHART_COLOR_CANDLE_BEAR,0,result))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
    }
    //--- Geben den Wert der Eigenschaft zurück
    return((color)result);
};

//+-----+
//| Die Funktion setzt die Farbe des Körpers von Bär-Kerze. |
//+-----+
bool ChartBearColorSet(const color clr,const long chart_ID=0)
{
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- setzen die Farbe des Körpers von Bär-Kerze
    if(!ChartSetInteger(chart_ID,CHART_COLOR_CANDLE_BEAR,clr))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
        return(false);
    }
    //--- die erfolgreiche Umsetzung
    return(true);
};

```

- **CHART\_COLOR\_BID** - Farbe der Linie des Bid-Preises.

```

//+-----+
//| Die Funktion erhält die Farbe der Bid-Linie. |
//+-----+
color ChartBidColorGet(const long chart_ID=0)
{
    //--- Bereiten wir eine Variable, um die Farbe zu erhalten
    long result=clrNONE;
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- erhalten die Farbe der Bid-Linie
    if(!ChartGetInteger(chart_ID,CHART_COLOR_BID,0,result))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
    }
    //--- Geben den Wert der Eigenschaft zurück

```

```

    return((color)result);
; }
//+-----+
//| Die Funktion setzt die Farbe der Bid-Linie. |
//+-----+
bool ChartBidColorSet(const color clr,const long chart_ID=0)
{
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- setzen die Farbe der Bid-Linie
    if(!ChartSetInteger(chart_ID,CHART_COLOR_BID,clr))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
        return(false);
    }
; }
//--- die erfolgreiche Umsetzung
return(true);
; }

```

- CHART\_COLOR\_ASK - Farbe der Linie des Ask-Preises.

```

//+-----+
//| Die Funktion erhält die Farbe der Ask-Linie. |
//+-----+
color ChartAskColorGet(const long chart_ID=0)
{
    //--- Bereiten wir eine Variable, um die Farbe zu erhalten
    long result=clrNONE;
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- erhalten die Farbe der Ask-Linie
    if(!ChartGetInteger(chart_ID,CHART_COLOR_ASK,0,result))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
    }
; }
//--- Geben den Wert der Eigenschaft zurück
return((color)result);
; }
//+-----+
//| Die Funktion setzt die Farbe der Ask-Linie. |
//+-----+
bool ChartAskColorSet(const color clr,const long chart_ID=0)
{
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- setzen die Farbe der Ask-Linie

```



```

    if(!ChartSetInteger(chart_ID,CHART_COLOR_ASK,clr))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
};

```

- **CHART\_COLOR\_LAST** - Farbe der Linie von letzten Transaktion (Last).

```

//+-----+
//| Die Funktion erhält die Farbe der Linie von letzten Transaktion (Last). |
//+-----+
color ChartLastColorGet(const long chart_ID=0)
{
    //--- Bereiten wir eine Variable, um die Farbe zu erhalten
    long result=clrNONE;
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- erhalten die Farbe der Linie von letzten Transaktion (Last)
    if(!ChartGetInteger(chart_ID,CHART_COLOR_LAST,0,result))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = ",GetLastError());
    }
    //--- Geben den Wert der Eigenschaft zurück
    return((color)result);
};
//+-----+
//| Die Funktion setzt die Farbe der Linie von letzten |
//| Transaktion. |
//+-----+
bool ChartLastColorSet(const color clr,const long chart_ID=0)
{
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- setzen die Farbe der Linie von letzten Transaktion (Last)
    if(!ChartSetInteger(chart_ID,CHART_COLOR_LAST,clr))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = ",GetLastError());
        return(false);
    }
    //--- die erfolgreiche Umsetzung
    return(true);
};

```

- **CHART\_COLOR\_STOP\_LEVEL** - Farbe von Stop-Orders (Stop Loss und Take Profit).

```
//+-----+
//| Die Funktion bekommt die Farbe von Stop Loss und Take Profit.      |
//+-----+
color ChartStopLevelColorGet(const long chart_ID=0)
{
//--- Bereiten wir eine Variable, um die Farbe zu erhalten
    long result=clrNONE;
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- erhalten die Farbe von Stop Loss und Take Profit.
    if(!ChartGetInteger(chart_ID,CHART_COLOR_STOP_LEVEL,0,result))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
    }
//--- Geben den Wert der Eigenschaft zurück
    return((color)result);
;}

//+-----+
//| Die Funktion setzt die Farbe von Stop Loss und Take Profit.      |
//+-----+
bool ChartStopLevelColorSet(const color clr,const long chart_ID=0)
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- setzen die Farbe von Stop Loss und Take Profit.
    if(!ChartSetInteger(chart_ID,CHART_COLOR_STOP_LEVEL,clr))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
;}
```

- **CHART\_SHOW\_TRADE\_LEVELS** - Eigenschaft der Anzeige von Handelsstufen auf dem Chart (Stufen der offenen Positionen, Stop Loss, Take Profit und Pending Orders).

```
//+-----+
//| Die Funktion ermittelt, ob die Handelsstufen auf dem Chart angezeigt werden. |
//+-----+
bool ChartShowTradeLevelsGet(bool &result,const long chart_ID=0)
{
;
```

```

//--- Bereiten wir eine Variable, um den Wert der Eigenschaft zu erhalten
    long value;
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Erhalten wir den Wert der Eigenschaft
    if(!ChartGetInteger(chart_ID, CHART_SHOW_TRADE_LEVELS, 0, value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", "Error Code = ", GetLastError());
        return(false);
    }
//--- speichern den Wert der Chat-Eigenschaft in der Variable
    result=value;
//--- die erfolgreiche Umsetzung
    return(true);
; }

//+-----+
//| Die Funktion aktiviert/deaktiviert die Anzeige der Handelsstufen. |
//+-----+
bool ChartShowTradeLevelsSet(const bool value, const long chart_ID=0)
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Geben wir den Wert der Eigenschaft ein
    if(!ChartSetInteger(chart_ID, CHART_SHOW_TRADE_LEVELS, 0, value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", "Error Code = ", GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
; }

```

- **CHART\_DRAG\_TRADE\_LEVELS** - Eigenschaft von Ziehen der Handelsstufen auf dem Chart mit dem Maus.

```

//+-----+
//| Die Funktion ermittelt, ob Ziehen von Handelsstufen auf dem Chart |
//| mit dem Maus erlaubt ist. |
//+-----+
bool ChartDragTradeLevelsGet(bool &result, const long chart_ID=0)
{
//--- Bereiten wir eine Variable, um den Wert der Eigenschaft zu erhalten
    long value;
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Erhalten wir den Wert der Eigenschaft

```

```

    if(!ChartGetInteger(chart_ID,CHART_DRAG_TRADE_LEVELS,0,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = ",GetLastError());
        return(false);
    }
    //--- speichern den Wert der Chat-Eigenschaft in der Variable
    result=value;
    //--- die erfolgreiche Umsetzung
    return(true);
};

//+-----+
//| Die Funktion aktiviert/deaktiviert Ziehen der Handelsstufen |
//| auf dem Chart mit dem Maus.                                |
//+-----+
bool ChartDragTradeLevelsSet(const bool value,const long chart_ID=0)
{
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Geben wir den Wert der Eigenschaft ein
    if(!ChartSetInteger(chart_ID,CHART_DRAG_TRADE_LEVELS,0,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = ",GetLastError());
        return(false);
    }
    //--- die erfolgreiche Umsetzung
    return(true);
};

```

- **CHART\_SHOW\_DATE\_SCALE** - Eigenschaft der Anzeige der Zeitskala auf dem Chart.

```

//+-----+
//| Die Funktion ermittelt, ob die Zeitskala auf dem Chart angezeigt wird. |
//+-----+
bool ChartShowDateScaleGet(bool &result,const long chart_ID=0)
{
    //--- Bereiten wir eine Variable, um den Wert der Eigenschaft zu erhalten
    long value;
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Erhalten wir den Wert der Eigenschaft
    if(!ChartGetInteger(chart_ID,CHART_SHOW_DATE_SCALE,0,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = ",GetLastError());
        return(false);
    }
};

```

```

//--- speichern den Wert der Chat-Eigenschaft in der Variable
    result=value;
//--- die erfolgreiche Umsetzung
    return(true);
; }
//+-----+
//| Die Funktion aktiviert/deaktiviert Anzeige der Zeitskala auf dem |
//| Chart.                                                         |
//+-----+
bool ChartShowDateScaleSet(const bool value,const long chart_ID=0)
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Geben wir den Wert der Eigenschaft ein
    if(!ChartSetInteger(chart_ID,CHART_SHOW_DATE_SCALE,0,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
; }

```

• **CHART\_SHOW\_PRICE\_SCALE** - Eigenschaft der Anzeige der Preiseskala auf dem Chart..

```

//+-----+
//| Die Funktion ermittelt, ob die Preiseskala auf dem Chart angezeigt wird. |
//+-----+
bool ChartShowPriceScaleGet(bool &result,const long chart_ID=0)
{
//--- Bereiten wir eine Variable, um den Wert der Eigenschaft zu erhalten
    long value;
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Erhalten wir den Wert der Eigenschaft
    if(!ChartGetInteger(chart_ID,CHART_SHOW_PRICE_SCALE,0,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
        return(false);
    }
//--- speichern den Wert der Chat-Eigenschaft in der Variable
    result=value;
//--- die erfolgreiche Umsetzung
    return(true);
; }
//+-----+

```

```

//| Die Funktion aktiviert/deaktiviert Anzeige der Preiseskala auf dem |
//| Chart. |
//+-----+
bool ChartShowPriceScaleSet(const bool value,const long chart_ID=0)
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Geben wir den Wert der Eigenschaft ein
    if(!ChartSetInteger(chart_ID,CHART_SHOW_PRICE_SCALE,0,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
};

```

- **CHART\_SHOW\_ONE\_CLICK** - отображение на графике панели быстрой торговли (опция "Торговля одним кликом").

```

//+-----+
//| Функция определяет, отображается ли на графике |
//| панель быстрой торговли ("Торговля одним кликом") |
//+-----+
bool ChartShowOneClickPanelGet(bool &result,const long chart_ID=0)
{
//--- подготовим переменную для получения значения свойства
    long value;
//--- сбросим значение ошибки
    ResetLastError();
//--- получим значение свойства
    if(!ChartGetInteger(chart_ID,CHART_SHOW_ONE_CLICK,0,value))
    {
        //--- выведем сообщение об ошибке в журнал "Эксперты"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
        return(false);
    }
//--- запомним в переменную значение свойства графика
    result=value;
//--- успешное выполнение
    return(true);
}
//+-----+
//| Функция включает/выключает режим отображения |
//| панели быстрой торговли на графике |
//+-----+
bool ChartShowOneClickPanelSet(const bool value,const long chart_ID=0)

```

```

{
//--- сбросим значение ошибки
    ResetLastError();
//--- установим значение свойства
    if(!ChartSetInteger(chart_ID,CHART_SHOW_ONE_CLICK,0,value))
    {
        //--- выведем сообщение об ошибке в журнал "Эксперты"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
        return(false);
    }
//--- успешное выполнение
    return(true);
}

```

- **CHART\_SHIFT\_SIZE** - Verschiebung des Null-Baren vom rechten Rand in Prozent.

```

//+-----+
//| Die Funktion erhält die Größe der Verschiebung des Null-Baren vom rechten Rand
//| des Charts in Prozent (von 10% bis 50%). |
//+-----+
double ChartShiftSizeGet(const long chart_ID=0)
{
//--- Bereiten wir eine Variable, um das Ergebnis zu erhalten
    double result=EMPTY_VALUE;
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Erhalten wir den Wert der Eigenschaft
    if(!ChartGetDouble(chart_ID,CHART_SHIFT_SIZE,0,result))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
    }
//--- Geben den Wert der Eigenschaft zurück
    return(result);
; }
//+-----+
//| Die Funktion setzt die Größe der Verschiebung des Null-Baren vom rechten Rand
//| des Charts in Prozent (von 10% bis 50%). Um den Modus der Verschiebung |
//| zu aktivieren, setzen Sie den Wert von CHART_SHIFT zu |
//| true. |
//+-----+
bool ChartShiftSizeSet(const double value,const long chart_ID=0)
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Geben wir den Wert der Eigenschaft ein
    if(!ChartSetDouble(chart_ID,CHART_SHIFT_SIZE,value))
    {

```

```

    //--- Schreiben die Fehlermeldung in den Log "Experten"
    Print(__FUNCTION__+"", Error Code = "", GetLastError());
    return(false);
};
//--- die erfolgreiche Umsetzung
return(true);
};

```

- **CHART\_FIXED\_POSITION** - Position der festen Position des Chart vom linken Rand als Prozentsatz.

```

//+-----+
//| Die Funktion erhält die Position der festen Position des Chart vom |
//| linken Rand als Prozentsatz. |
//+-----+
double ChartFixedPositionGet(const long chart_ID=0)
{
    //--- Bereiten wir eine Variable, um das Ergebnis zu erhalten
    double result=EMPTY_VALUE;
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Erhalten wir den Wert der Eigenschaft
    if(!ChartGetDouble(chart_ID, CHART_FIXED_POSITION, 0, result))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = "", GetLastError());
    };
    //--- Geben den Wert der Eigenschaft zurück
    return(result);
};
//+-----+
//| Die Funktion setzt die Position der festen Position des Chart vom |
//| linken Rand als Prozentsatz. Um die Position der feste Position auf dem Chart |
//| zu sehen, müssen Sie zunächst den Wert der Eigenschaft |
//| CHART_AUTOSCROLL auf false setzen. |
//+-----+
bool ChartFixedPositionSet(const double value, const long chart_ID=0)
{
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Geben wir den Wert der Eigenschaft ein
    if(!ChartSetDouble(chart_ID, CHART_FIXED_POSITION, value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = "", GetLastError());
        return(false);
    };
    //--- die erfolgreiche Umsetzung
    return(true);
};

```



```
;}
```

- **CHART\_FIXED\_MAX** - die Eigenschaft des festen Maximalwertes des Charts.

```
//+-----+
//| Die Funktion erhält den festen Maximalwert des Charts. |
//+-----+
double ChartFixedMaxGet(const long chart_ID=0)
{
//--- Bereiten wir eine Variable, um das Ergebnis zu erhalten
    double result=EMPTY_VALUE;
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Erhalten wir den Wert der Eigenschaft
    if(!ChartGetDouble(chart_ID,CHART_FIXED_MAX,0,result))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = "",GetLastError());
    }
//--- Geben den Wert der Eigenschaft zurück
    return(result);
};

//+-----+
//| Die Funktion setzt den festen Maximalwert des Charts. |
//| Um den Wert dieser Eigenschaft zu ändern, setzen Sie |
//| zunächst den Wert der Eigenschaft CHART_SCALEFIX auf |
//| true. |
//+-----+
bool ChartFixedMaxSet(const double value,const long chart_ID=0)
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Geben wir den Wert der Eigenschaft ein
    if(!ChartSetDouble(chart_ID,CHART_FIXED_MAX,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = "",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
};
```

- **CHART\_FIXED\_MIN** - Eigenschaft des festen Minimalwertes des Charts.

```
//+-----+
//| Die Funktion erhält den festen Minimalwert des Charts. |
```

```
//+-----+
double ChartFixedMinGet(const long chart_ID=0)
{
//--- Bereiten wir eine Variable, um das Ergebnis zu erhalten
    double result=EMPTY_VALUE;
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Erhalten wir den Wert der Eigenschaft
    if(!ChartGetDouble(chart_ID,CHART_FIXED_MIN,0,result))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = "",GetLastError());
    }
//--- Geben den Wert der Eigenschaft zurück
    return(result);
;}

//+-----+
//| Die Funktion setzt den festen Minimalwert des Charts. |
//| Um den Wert dieser Eigenschaft zu ändern, setzen Sie |
//| zunächst den Wert der Eigenschaft CHART_SCALEFIX auf |
//| true. |
//+-----+
bool ChartFixedMinSet(const double value,const long chart_ID=0)
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Geben wir den Wert der Eigenschaft ein
    if(!ChartSetDouble(chart_ID,CHART_FIXED_MIN,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = "",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
;}

```

- **CHART\_POINTS\_PER\_BAR** - Skalenwert als Punkten pro Bar.

```
//+-----+
//| Die Funktion erhält den Skalenwert als Punkten pro Bar. |
//+-----+
double ChartPointsPerBarGet(const long chart_ID=0)
{
//--- Bereiten wir eine Variable, um das Ergebnis zu erhalten
    double result=EMPTY_VALUE;
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();

```

```

//--- Erhalten wir den Wert der Eigenschaft
if(!ChartGetDouble(chart_ID,CHART_POINTS_PER_BAR,0,result))
{
    //--- Schreiben die Fehlermeldung in den Log "Experten"
    Print(__FUNCTION__+"", Error Code = "",GetLastError());
};
//--- Geben den Wert der Eigenschaft zurück
return(result);
};
//+-----+
//| Die Funktion setzt den Skalenwert als Punkten pro Bar.
//| Um das Ergebnis von Veränderungen des Wertes dieser |
//| Eigenschaft zu sehen, setzen Sie zunächst den Wert von |
//| CHART_SCALE_PT_PER_BAR auf true. |
//+-----+
bool ChartPointsPerBarSet(const double value,const long chart_ID=0)
{
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Geben wir den Wert der Eigenschaft ein
    if(!ChartSetDouble(chart_ID,CHART_POINTS_PER_BAR,value))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = "",GetLastError());
        return(false);
    };
    //--- die erfolgreiche Umsetzung
    return(true);
};

```

- **CHART\_PRICE\_MIN** - gibt den Minimalwert des Charts zurück.

```

//+-----+
//| Die Funktion erhält den Minimalwert des Charts im Hauptfenster oder |
//| Untenfenster. |
//+-----+
double ChartPriceMin(const long chart_ID=0,const int sub_window=0)
{
    //--- Bereiten wir eine Variable, um das Ergebnis zu erhalten
    double result=EMPTY_VALUE;
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Erhalten wir den Wert der Eigenschaft
    if(!ChartGetDouble(chart_ID,CHART_PRICE_MIN,sub_window,result))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = "",GetLastError());
    };
};

```

```
//--- Geben den Wert der Eigenschaft zurück
    return(result);
;}
```

- **CHART\_PRICE\_MAX** - gibt den Maximalwert des Charts zurück.

```
//+-----+
//| Die Funktion erhält den Maximalwert des Charts im Hauptfenster oder |
//| Untenfenster. |
//+-----+
double ChartPriceMax(const long chart_ID=0,const int sub_window=0)
{
    //--- Bereiten wir eine Variable, um das Ergebnis zu erhalten
    double result=EMPTY_VALUE;
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Erhalten wir den Wert der Eigenschaft
    if(!ChartGetDouble(chart_ID,CHART_PRICE_MAX,sub_window,result))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
    }
    //--- Geben den Wert der Eigenschaft zurück
    return(result);
;}
```

- **CHART\_COMMENT** - Text der Kommentar auf dem Chart.

```
//+-----+
//| Die Funktion bekommt den Kommentartext in der linken oberen Ecke des Charts. |
//+-----+
bool ChartCommentGet(string &result,const long chart_ID=0)
{
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Erhalten wir den Wert der Eigenschaft
    if(!ChartGetString(chart_ID,CHART_COMMENT,result))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+" Error Code = ",GetLastError());
        return(false);
    }
    //--- die erfolgreiche Umsetzung
    return(true);
;}
```

```
//+-----+
//| Die Funktion bekommt den Kommentartext in der linken oberen Ecke |
```

```

//| des Charts. |
//+-----+
bool ChartCommentSet(const string str,const long chart_ID=0)
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Geben wir den Wert der Eigenschaft ein
    if(!ChartSetString(chart_ID,CHART_COMMENT,str))
    {
        //--- Schreiben die Fehlermeldung in den Log "Experten"
        Print(__FUNCTION__+"", Error Code = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
};

```

### Panel für die Charteigenschaften

```

//--- Schließen Sie das Control-Bibliothek
#include <ChartObjects\ChartObjectsTxtControls.mqh>
//--- Vordefinierte Konstanten
#define X_PROPERTY_NAME_1    10  // X-Koordinate der Eigenschaft Name in der ersten Spalte
#define X_PROPERTY_VALUE_1  225 // X-Koordinaten des Eigenschaftswertes in der ersten Spalte
#define X_PROPERTY_NAME_2    345 // X-Koordinate der Eigenschaft Name in der zweiten Spalte
#define X_PROPERTY_VALUE_2  550 // X-Koordinaten des Eigenschaftswertes in der zweiten Spalte
#define X_BUTTON_1           285 // X-Koordinate der Taste in der ersten Spalte
#define X_BUTTON_2           700 // X-Koordinate der Taste in der zweiten Spalte
#define Y_PROPERTY_1         30  // Y-Koordinate des Anfangspunktes von der ersten Spalte
#define Y_PROPERTY_2         286 // Y-Koordinate des Anfangspunktes von der dritten Spalte
#define Y_DISTANCE           16  // Der y-axiale Abstand zwischen den Zeilen
#define LAST_PROPERTY_NUMBER 111 // Nummer der letzten grafischen Eigenschaften
//--- Input-Parameter
input color InpFirstColor=clrDodgerBlue; // Farbe der ungeraden Zeilen
input color InpSecondColor=clrGoldenrod; // Farbe der geraden Zeilen
//--- Variablen und Arrays
CChartObjectLabel ExtLabelsName[]; // Label, um die Namen der Eigenschaften anzuzeigen
CChartObjectLabel ExtLabelsValue[]; // Label, um die Namen der Werte anzuzeigen
CChartObjectButton ExtButtons[]; // Tasten
int ExtNumbers[]; // Indizes der Eigenschaften
string ExtNames[]; // Namen der Eigenschaften
uchar ExtDataTypes[]; // Datentypen von Eigenschaften (integer, double, etc.)
uint ExtGroupTypes[]; // Array, das Informationen über die Zugehörigkeit der Eigenschaften enthält
uchar ExtDrawTypes[]; // Array, das Informationen über die Methode der Zeichnung enthält
double ExtMaxValue[]; // Maximale Werte der Eigenschaften, die sie in der ersten Spalte annehmen
double ExtMinValue[]; // Minimale Werte der Eigenschaften, die sie in der ersten Spalte annehmen
double ExtStep[]; // Schritte zum Ändern der Eigenschaften
int ExtCount; // Gesamtzahl der Eigenschaften

```

```

color          ExtColors[2];      // Array von Farben, um Zeilen anzuzeigen
string         ExtComments[2];    // Array von Kommentaren (für die Eigenschaft CH2
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- Kommentar auf dem Chart anzeigen
    Comment("SomeComment");
//--- Farben in das Array speichern, um später zwischen ihnen umzuschalten
    ExtColors[0]=InpFirstColor;
    ExtColors[1]=InpSecondColor;
//--- Kommentare in das Array speichern, um später zwischen ihnen umzuschalten
    ExtComments[0]="FirstComment";
    ExtComments[1]="SecondComment";
//--- Bedienfeld vorbereiten und anzeigen
    if(!PrepareControls())
        return(INIT_FAILED);
//--- die erfolgreiche Umsetzung
    return(INIT_SUCCEEDED);
};
//+-----+
//| Deinitialization function of the expert |
//+-----+
void OnDeinit(const int reason)
{
//--- Den Text des Kommentars aus dem Chart löschen
    Comment("");
};
//+-----+
//| Chart Event Handler |
//+-----+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
{
//--- Überprüfen das Ereignis von Klick auf das Objekt es Charts
    if(id==CHARTEVENT_OBJECT_CLICK)
    {
        //--- Aufteilen den Namen des Objekts auf Abscheider
        string obj_name[];
        StringSplit(sparam, '_', obj_name);
        //--- Überprüfen ob das Objekt eine Taste ist
        if(obj_name[0]=="Button")
        {
            //--- Index der Taste erhalten
            int index=(int)StringToInteger(obj_name[1]);
            //--- Taste in nicht gedrückte Zustand setzen

```

```

        ExtButtons[index].State(false);
        //--- Den neuen Wert der Eigenschaft je nach ihr Typ setzen
        if(ExtDataTypes[index]=='I')
            ChangeIntegerProperty(index);
        if(ExtDataTypes[index]=='D')
            ChangeDoubleProperty(index);
        if(ExtDataTypes[index]=='S')
            ChangeStringProperty(index);
    };
};

//--- Eigenschaftswerten neu zeichnen
RedrawProperties();
ChartRedraw();
};

//+-----+
//| Ändern die integrale Eigenschaft des Charts |
//+-----+
void ChangeIntegerProperty(const int index)
{
    //--- Erhalten wir den aktuellen Wert der Eigenschaft
    long value=ChartGetInteger(0,(ENUM_CHART_PROPERTY_INTEGER)ExtNumbers[index]);
    //--- Definieren den folgenden Wert der Eigenschaft
    switch(ExtDrawTypes[index])
    {
        case 'C':
            value=GetNextColor((color)value);
            break;
        default:
            value=(long)GetNextValue((double)value,index);
            break;
    };
};

//--- Geben wir den aktuellen Wert der Eigenschaft ein
ChartSetInteger(0,(ENUM_CHART_PROPERTY_INTEGER)ExtNumbers[index],0,value);
};

//+-----+
//| Ändern die echte Eigenschaft des Charts |
//+-----+
void ChangeDoubleProperty(const int index)
{
    //--- Erhalten wir den aktuellen Wert der Eigenschaft
    double value=ChartGetDouble(0,(ENUM_CHART_PROPERTY_DOUBLE)ExtNumbers[index]);
    //--- Definieren den folgenden Wert der Eigenschaft
    value=GetNextValue(value,index);
    //--- Geben wir den aktuellen Wert der Eigenschaft ein
    ChartSetDouble(0,(ENUM_CHART_PROPERTY_DOUBLE)ExtNumbers[index],value);
};

//+-----+
//| Ändern die String-Eigenschaft des Charts |
//+-----+

```

```

void ChangeStringProperty(const int index)
{
    //--- Statische Variable um Kommentare ExtComments innerhalb des Arrays zu wechseln
    static uint comment_index=1;
    //--- Ändern Index für einen weiteren Kommentar
    comment_index=1-comment_index;
    //--- Geben wir den aktuellen Wert der Eigenschaft ein
    ChartSetString(0, (ENUM_CHART_PROPERTY_STRING)ExtNumbers[index],ExtComments[comment_
; }
//+-----+
//| Identifizierung des nächsten Wertes der Eigenschaft |
//+-----+
double GetNextValue(const double value,const int index)
{
    if (value+ExtStep[index]<=ExtMaxValue[index])
        return (value+ExtStep[index]);
    else
        return (ExtMinValue[index]);
; }
//+-----+
//| Erhalten die nächste Farbe für die Eigenschaft von Typ color |
//+-----+
color GetNextColor(const color clr)
{
    //--- Den nächsten Farbwert zurückgeben
    switch (clr)
    {
        case clrWhite: return (clrRed);
        case clrRed:   return (clrGreen);
        case clrGreen: return (clrBlue);
        case clrBlue:  return (clrBlack);
        default:       return (clrWhite);
    ; }
; }
//+-----+
//| Eigenschaftswerten neu zeichnen |
//+-----+
void RedrawProperties(void)
{
    //--- Text des Eigenschaftswertes
    string text;
    long   value;
    //--- Zyklus über die Anzahl der Eigenschaften
    for (int i=0; i<ExtCount; i++)
    {
        text="";
        switch (ExtDataTypes[i])
        {
            case 'I':

```



```

//--- Erhalten wir den aktuellen Wert der Eigenschaft
if(!ChartGetInteger(0,(ENUM_CHART_PROPERTY_INTEGER)ExtNumbers[i],0,value))
break;
//--- Text der Integer-Eigenschaft
switch(ExtDrawTypes[i])
{
//--- Farbeigenschaft
case 'C':
text=(string)((color)value);
break;
//--- Boolesche Eigenschaft
case 'B':
text=(string)((bool)value);
break;
//--- Eigenschaft von ENUM_CHART_MODE
case 'M':
text=EnumToString((ENUM_CHART_MODE)value);
break;
//--- Eigenschaft von ENUM_CHART_VOLUME_MODE
case 'V':
text=EnumToString((ENUM_CHART_VOLUME_MODE)value);
break;
//--- Nummer er Typ int
default:
text=IntegerToString(value);
break;
};
break;
case 'D':
//--- Text der echten Eigenschaft
text=DoubleToString(ChartGetDouble(0,(ENUM_CHART_PROPERTY_DOUBLE)ExtNumbers[i],0,value));
break;
case 'S':
//--- Text der String-Eigenschaft
text=ChartGetString(0,(ENUM_CHART_PROPERTY_STRING)ExtNumbers[i]);
break;
};
//--- Anzeige der Wert der Eigenschaft
ExtLabelsValue[i].Description(text);
};
};
//+-----+
//| Erstellung einer Platte,um die Eigenschaften des Charts zu kontrollieren
//+-----+
bool PrepareControls(void)
{
//--- Reservieren Speicher für Arrays mit Reserve
MemoryAllocation(LAST_PROPERTY_NUMBER+1);
//--- Variablen

```

```

int i=0;      // Zyklus-Variable
int col_1=0; // Anzahl der Eigenschaften in der erste Spalte
int col_2=0; // Anzahl der Eigenschaften in der zweite Spalte
int col_3=0; // Anzahl der Eigenschaften in der dritte Spalte
//--- Die aktuelle Anzahl von Eigenschaften ist 0
ExtCount=0;
//--- Suche nach Eigenschaften in Zyklus
while(i<=LAST_PROPERTY_NUMBER)
{
    //--- Den aktuellen Nummer der Eigenschaft speichern
    ExtNumbers[ExtCount]=i;
    //--- Den Wert der Zyklus-Eigenschaft erhöhen
    i++;
    //--- Prüfen, ob es eine Eigenschaft mit diesem Nummer gibt
    if(CheckNumber(ExtNumbers[ExtCount],ExtNames[ExtCount],ExtDataTypes[ExtCount],ExtDataValues[ExtCount])<0)
    {
        //--- Steuerelemente für Eigenschaften erstellen
        switch(ExtGroupTypes[ExtCount])
        {
            case 1:
                //--- ein Label und eine Schaltfläche für die Eigenschaft erstellen
                if(!ShowProperty(ExtCount,0,X_PROPERTY_NAME_1,X_PROPERTY_VALUE_1,X_BUTTON_1))
                    return(false);
                //--- Die Anzahl der Elemente in der ersten Spalte hat erhöht
                col_1++;
                break;
            case 2:
                //--- ein Label und eine Schaltfläche für die Eigenschaft erstellen
                if(!ShowProperty(ExtCount,1,X_PROPERTY_NAME_2,X_PROPERTY_VALUE_2,X_BUTTON_2))
                    return(false);
                //--- Die Anzahl der Elemente in der zweite Spalte hat erhöht
                col_2++;
                break;
            case 3:
                //--- Nur Label für Eigenschaften erstellen
                if(!ShowProperty(ExtCount,2,X_PROPERTY_NAME_2,X_PROPERTY_VALUE_2,0,X_BUTTON_3))
                    return(false);
                //--- Die Anzahl der Elemente in der dritten Spalte hat erhöht
                col_3++;
                break;
        }
        //--- Den maximalen und minimalen Wert und den Schritt definieren
        GetMaxMinStep(ExtNumbers[ExtCount],ExtMaxValue[ExtCount],ExtMinValue[ExtCount],ExtStep[ExtCount]);
        //--- Erhöhung der Anzahl der Eigenschaften
        ExtCount++;
    }
}
//--- Freien wir den Speicher, der durch Arrays nicht verwendet wird
MemoryAllocation(ExtCount);

```

```

//--- Eigenschaftswerten neu zeichnen
RedrawProperties();
ChartRedraw();
//--- die erfolgreiche Umsetzung
return(true);
;}

//+-----+
//| Speicher für Arrays reservieren |
//+-----+
void MemoryAllocation(const int size)
{
    ArrayResize(ExtLabelsName,size);
    ArrayResize(ExtLabelsValue,size);
    ArrayResize(ExtButtons,size);
    ArrayResize(ExtNumbers,size);
    ArrayResize(ExtNames,size);
    ArrayResize(ExtDataTypes,size);
    ArrayResize(ExtGroupTypes,size);
    ArrayResize(ExtDrawTypes,size);
    ArrayResize(ExtMaxValue,size);
    ArrayResize(ExtMinValue,size);
    ArrayResize(ExtStep,size);
;}

//+-----+
//| Prüfen ob der Index der Eigenschaft zu einem der Aufzählungen |
//| ENUM_CHART_PROPERTIES gehört |
//+-----+
bool CheckNumber(const int ind,string &name,uchar &data_type,uint &group_type,uchar &
{
//--- Prüfen, ob die Eigenschaft eine ganze Zahl ist
ResetLastError();
name=EnumToString((ENUM_CHART_PROPERTY_INTEGER)ind);
if(_LastError==0)
{
    data_type='I'; // Eigenschaft aus ENUM_CHART_PROPERTY_INTEGER
    GetTypes(ind,group_type,draw_type); // Parameter von Anzeige der Eigenschaften
    return(true);
;}

//--- Prüfen, ob die Eigenschaft echt ist
ResetLastError();
name=EnumToString((ENUM_CHART_PROPERTY_DOUBLE)ind);
if(_LastError==0)
{
    data_type='D'; // Eigenschaft aus Aufzählung ENUM_CHART_PROPERTY_DOUBLE
    GetTypes(ind,group_type,draw_type); // Parameter von Anzeige der Eigenschaften
    return(true);
;}

//--- Prüfen, ob die Eigenschaft String ist
ResetLastError();

```

```

name=EnumToString((ENUM_CHART_PROPERTY_STRING)ind);
if(_LastError==0)
{
    data_type='S'; // Eigenschaft aus Aufzählung ENUM_CHART_PRO
    GetTypes(ind,group_type,draw_type); // Parameter von Anzeige der Eigenschaften d
    return(true);
};
//--- die Eigenschaft gehört zu keiner Aufzählung
return(false);
};
//+-----+
//| Definieren, in welcher Gruppe die Eigenschaft sein sollte, |
//| und ihr Display-Typ |
//+-----+
void GetTypes(const int property_number,uint &group_type,uchar &draw_type)
{
    //--- Prüfen, ob die Eigenschaft zur dritten Grippe gehört
    //--- Eigenschaften der dritten Gruppe werden in der zweite Spalte von CHART_BRING_TO_
    if(CheckThirdGroup(property_number,group_type,draw_type))
        return;
    //--- Prüfen, ob die Eigenschaft zur zweiten Grippe gehört
    //--- Eigenschaften der dritten Gruppe werden in der zweite Spalte von Anfang angezeigt
    if(CheckSecondGroup(property_number,group_type,draw_type))
        return;
    //--- Wenn Sie hier sind, bedeutet es, das die Eigenschaft zur ersten Gruppe (erste Sp
    CheckFirstGroup(property_number,group_type,draw_type);
};
//+-----+
//| Die Funktion überprüft, ob die Eigenschaft zur dritten Gruppe gehört, und |
//| wenn ja, bestimmt den Display-Typ |
//+-----+
bool CheckThirdGroup(const int property_number,uint &group_type,uchar &draw_type)
{
    //--- Prüfen, ob die Eigenschaft zur dritten Grippe gehört
    switch(property_number)
    {
        //--- Boolesche Eigenschaften
        case CHART_IS_OBJECT:
        case CHART_WINDOW_IS_VISIBLE:
            draw_type='B';
            break;
        //--- Integer Eigenschaften
        case CHART_VISIBLE_BARS:
        case CHART_WINDOWS_TOTAL:
        case CHART_WINDOW_HANDLE:
        case CHART_WINDOW_YDISTANCE:
        case CHART_FIRST_VISIBLE_BAR:
        case CHART_WIDTH_IN_BARS:
        case CHART_WIDTH_IN_PIXELS:

```

```

        draw_type='I';
        break;
        //--- echte Eigenschaften
    case CHART_PRICE_MIN:
    case CHART_PRICE_MAX:
        draw_type='D';
        break;
        //--- In der Tat ist diese Eigenschaft eine Instruktion den Chat an der Spitze
        //--- Für dieses Panel ist ihre Anwendung nicht notwendig, da das Fenster ist
        //--- über alle anderen Fenster
    case CHART_BRING_TO_TOP:
        draw_type=' ';
        break;
        //--- Die Eigenschaft gehört nicht zur dritten Gruppe
    default:
        return(false);
    };
    //--- Die Eigenschaft gehört zur dritten Gruppe
    group_type=3;
    return(true);
};
//+-----+
//| Die Funktion überprüft, ob die Eigenschaft zur zweiten Gruppe gehört, und      |
//| wenn ja, bestimmt den Display-Typ                                             |
//+-----+
bool CheckSecondGroup(const int property_number,uint &group_type,uchar &draw_type)
{
    //--- Prüfen, ob die Eigenschaft zur zweiten Gruppe gehört
    switch(property_number)
    {
        //--- Eigenschaft von Typ ENUM_CHART_MODE
    case CHART_MODE:
        draw_type='M';
        break;
        //--- Eigenschaft von Typ ENUM_CHART_VOLUME_MODE
    case CHART_SHOW_VOLUMES:
        draw_type='V';
        break;
        //--- String-Eigenschaft
    case CHART_COMMENT:
        draw_type='S';
        break;
        //--- Eigenschaft der Farbe
    case CHART_COLOR_BACKGROUND:
    case CHART_COLOR_FOREGROUND:
    case CHART_COLOR_GRID:
    case CHART_COLOR_VOLUME:
    case CHART_COLOR_CHART_UP:
    case CHART_COLOR_CHART_DOWN:

```

```

        case CHART_COLOR_CHART_LINE:
        case CHART_COLOR_CANDLE_BULL:
        case CHART_COLOR_CANDLE_BEAR:
        case CHART_COLOR_BID:
        case CHART_COLOR_ASK:
        case CHART_COLOR_LAST:
        case CHART_COLOR_STOP_LEVEL:
            draw_type='C';
            break;
        //--- Die Eigenschaft gehört nicht zur zweiten Gruppe
    default:
        return(false);
    };
//--- Die Eigenschaft gehört zur zweiten Gruppe
    group_type=2;
    return(true);
};
//+-----+
//| Diese Funktion wird nur aufgerufen, wenn es bereits bekannt ist, |
//| dass die Eigenschaft nicht zur zweiten und dritten Gruppe von Eigenschaften gehört |
//+-----+
void CheckFirstGroup(const int property_number, uint &group_type, uchar &draw_type)
{
    //--- Die Eigenschaft gehört zur ersten Gruppe
    group_type=1;
    //--- Typ von Anzeige der Eigenschaft definieren
    switch(property_number)
    {
        //--- Integer Eigenschaften
        case CHART_SCALE:
        case CHART_HEIGHT_IN_PIXELS:
            draw_type='I';
            return;
        //--- echte Eigenschaften
        case CHART_SHIFT_SIZE:
        case CHART_FIXED_POSITION:
        case CHART_FIXED_MAX:
        case CHART_FIXED_MIN:
        case CHART_POINTS_PER_BAR:
            draw_type='D';
            return;
        //--- Nur Boolesche Eigenschaften
    default:
        draw_type='B';
        return;
    };
};
//+-----+
//| Ein Label und eine Schaltfläche für die Eigenschaft erstellen |

```

```

//+-----+
bool ShowProperty(const int ind,const int type,const int x1,const int x2,
                  const int xb,const int y,const bool btn)
{
//--- Statischer Array um innerhalb des Farbe-Arrays ExtColors zu wechseln
    static uint color_index[3]={1,1,1};
//--- Ändern Index für eine andere Farbe
    color_index[type]=1-color_index[type];
//--- Label und Taste (wenn btn=true) für die Eigenschaft anzeigen
    if(!LabelCreate(ExtLabelsName[ind],"name_"+(string)ind,ExtNames[ind],ExtColors[color_index[type]])
        return(false);
    if(!LabelCreate(ExtLabelsValue[ind],"value_"+(string)ind,"",ExtColors[color_index[type]])
        return(false);
    if(btn && !ButtonCreate(ExtButtons[ind],(string)ind,xb,y+1))
        return(false);
//--- die erfolgreiche Umsetzung
    return(true);
;}
//+-----+
//| Label erstellen |
//+-----+
bool LabelCreate(CChartObjectLabel &lbl,const string name,const string text,
                  const color clr,const int x,const int y)
{
    if(!lbl.Create(0,"Label_"+name,0,x,y)) return(false);
    if(!lbl.Description(text))             return(false);
    if(!lbl.FontSize(10))                   return(false);
    if(!lbl.Color(clr))                     return(false);
//--- die erfolgreiche Umsetzung
    return(true);
;}
//+-----+
//| Taste erstellen |
//+-----+
bool ButtonCreate(CChartObjectButton &btn,const string name,
                  const int x,const int y)
{
    if(!btn.Create(0,"Button_"+name,0,x,y,50,15)) return(false);
    if(!btn.Description("Next"))                  return(false);
    if(!btn.FontSize(10))                          return(false);
    if(!btn.Color(clrBlack))                        return(false);
    if(!btn.BackColor(clrWhite))                    return(false);
    if(!btn.BorderColor(clrBlack))                  return(false);
//--- die erfolgreiche Umsetzung
    return(true);
;}
//+-----+
//| Den maximalen und minimalen Wert und den Schritt eingeben |
//+-----+

```

```
void GetMaxMinStep(const int property_number, double &max, double &min, double &step)
{
    double value;
    ///--- Den neuen Wert der Eigenschaft je nach ihr Typ setzen
    switch(property_number)
    {
        case CHART_SCALE:
            max=5;
            min=0;
            step=1;
            break;
        case CHART_MODE:
        case CHART_SHOW_VOLUMES:
            max=2;
            min=0;
            step=1;
            break;
        case CHART_SHIFT_SIZE:
            max=50;
            min=10;
            step=2.5;
            break;
        case CHART_FIXED_POSITION:
            max=90;
            min=0;
            step=15;
            break;
        case CHART_POINTS_PER_BAR:
            max=19;
            min=1;
            step=3;
            break;
        case CHART_FIXED_MAX:
            value=ChartGetDouble(0, CHART_FIXED_MAX);
            max=value*1.25;
            min=value;
            step=value/32;
            break;
        case CHART_FIXED_MIN:
            value=ChartGetDouble(0, CHART_FIXED_MIN);
            max=value;
            min=value*0.75;
            step=value/32;
            break;
        case CHART_HEIGHT_IN_PIXELS:
            max=700;
            min=520;
            step=30;
            break;
    }
}
```



```
//--- Default values
default:
    max=1;
    min=0;
    step=1;
; }
; }
```

## Objektkonstanten

44 graphische Objekte werden vorausgesehen, die erzeugt und auf dem Preischart dargestellt werden können. Alle Konstanten für Arbeit mit Objekten werden in 9 Gruppen aufgeteilt:

- [Objekttypen](#) - Identifikatoren der graphischen Objekte;
- [Objekteigenschaften](#) - Arbeit mit Eigenschaften der graphischen Objekte;
- [Methoden von Objektsnap](#) - Konstanten der Objektpositionierung auf dem Chart;
- [Snapwinkel](#) - ermöglicht Winkel des Charts einzustellen, demgegenüber Positionieren des Objekts in Pixel erfolgt;
- [Sichtbarkeit der Objekte](#) - Festlegung von Timeframes, auf denen das Objekt sichtbar wird;
- [Größe von Elliott Wellen](#) - Abstufung der Wellenmarkierung;
- [Objekte von Gann](#) - Trendkonstanten für Gann Fan und Gann Grid;
- [Bestand von Web-Farben](#) - Konstanten der vorbestimmten Web-Farben; ;
- [Wingdings](#) - Symbolenkoden der Schrift Wingdings.

## Objekttypen

Bei der Erzeugung des graphischen Objekts durch die Funktion [ObjectCreate\(\)](#) muss man Typ des zu erzeugenden Objekts angeben, der einen der Werte der Aufzählung `ENUM_OBJECT` annehmen kann. Weitere Präzisierungen der [Eigenschaften](#) des zu erzeugenden Objekts ist durch die Funktion für die Arbeit mit [graphischen Objekten](#) möglich.

### ENUM\_OBJECT

| Identifikator                        |        | Beschreibung                   |
|--------------------------------------|--------|--------------------------------|
| <a href="#">OBJ_VLINE</a>            |        | Senkrechte Linie               |
| <a href="#">OBJ_HLINE</a>            | —      | horizontale Linie              |
| <a href="#">OBJ_TREND</a>            | /      | Trendlinie                     |
| <a href="#">OBJ_TRENDBYANGLE</a>     | ∠      | Trendlinie nach Winkel         |
| <a href="#">OBJ_CYCLES</a>           |        | Zykluslinien                   |
| <a href="#">OBJ_ARROWED_LINE</a>     | ↗      | Objekt "Linie mit einem Pfeil" |
| <a href="#">OBJ_CHANNEL</a>          | ↗↘     | Abstandsgleicher Kanal         |
| <a href="#">OBJ_STDDEVCHANNEL</a>    | ↗↘     | Kanal der Standardabweichung   |
| <a href="#">OBJ_REGRESSION</a>       | ↗↘     | Kanal der linearen Regression  |
| <a href="#">OBJ_PITCHFORK</a>        | ///    | Andrews' Pitchfork             |
| <a href="#">OBJ_GANNLIN</a>          | ↗<br>G | Gannlinie                      |
| <a href="#">OBJ_GANNFAN</a>          | ↗<br>G | Gannfan                        |
| <a href="#">OBJ_GANNGRID</a>         | ↗<br>G | Ganngrid                       |
| <a href="#">OBJ_FIBO</a>             | <br>F  | Fibonacci Retracement          |
| <a href="#">OBJ_FIBOTIMES</a>        | <br>F  | Zeitzonen Fibonacci            |
| <a href="#">OBJ_FIBOFAN</a>          | ↗<br>F | Fibonacci Fan                  |
| <a href="#">OBJ_FIBOARC</a>          | ↗<br>F | Fibonacci Arcs                 |
| <a href="#">OBJ_FIBOCHANNEL</a>      | ↗<br>F | Kanal Fibonacci                |
| <a href="#">OBJ_EXPANSION</a>        | ↗<br>F | Erweiterung Fibonacci          |
| <a href="#">OBJ_ELLIOTWAVE5</a>      | ↗<br>E | Elliott Motive Wave            |
| <a href="#">OBJ_ELLIOTWAVE3</a>      | ↗<br>F | Elliott Correction Wave        |
| <a href="#">OBJ_RECTANGLE</a>        | ■      | Rechteck                       |
| <a href="#">OBJ_TRIANGLE</a>         | ▲      | Dreieck                        |
| <a href="#">OBJ_ELLIPSE</a>          | ●      | Ellipse                        |
| <a href="#">OBJ_ARROW_THUMB_UP</a>   | 👍      | Zeichen "Gut" (Daumen oben)    |
| <a href="#">OBJ_ARROW_THUMB_DOWN</a> | 👎      | Zeichen "Schlecht" (Daumen     |

|                                              |                                                                                     |                                                                                   |
|----------------------------------------------|-------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------|
|                                              |                                                                                     | unten)                                                                            |
| <a href="#"><u>OBJ_ARROW_UP</u></a>          |    | Zeichen "Weiche oben"                                                             |
| <a href="#"><u>OBJ_ARROW_DOWN</u></a>        |    | Zeichen "Weiche unten"                                                            |
| <a href="#"><u>OBJ_ARROW_STOP</u></a>        |    | Zeichen "Stop"                                                                    |
| <a href="#"><u>OBJ_ARROW_CHECK</u></a>       |    | Zeichen "Kontrollmarke"                                                           |
| <a href="#"><u>OBJ_ARROW_LEFT_PRICE</u></a>  |    | linke Preismarke                                                                  |
| <a href="#"><u>OBJ_ARROW_RIGHT_PRICE</u></a> |    | Rechte Preismarke                                                                 |
| <a href="#"><u>OBJ_ARROW_BUY</u></a>         |    | Zeichen "Buy"                                                                     |
| <a href="#"><u>OBJ_ARROW_SELL</u></a>        |    | Zeichen "Sell"                                                                    |
| <a href="#"><u>OBJ_ARROW</u></a>             |    | Objekt "Weiche"                                                                   |
| <a href="#"><u>OBJ_TEXT</u></a>              |    | Objekt "Text"                                                                     |
| <a href="#"><u>OBJ_LABEL</u></a>             |    | Objekt "Textmarke"                                                                |
| <a href="#"><u>OBJ_BUTTON</u></a>            |    | Objekt "Schaltfläche"                                                             |
| <a href="#"><u>OBJ_CHART</u></a>             |    | Objekt "Chart"                                                                    |
| <a href="#"><u>OBJ_BITMAP</u></a>            |   | Objekt "Bild"                                                                     |
| <a href="#"><u>OBJ_BITMAP_LABEL</u></a>      |  | Objekt "graphische Marke"                                                         |
| <a href="#"><u>OBJ_EDIT</u></a>              |  | Objekt "Eingabefeld"                                                              |
| <a href="#"><u>OBJ_EVENT</u></a>             |  | Objekt "Ereignis", das einem Ereignis in der wirtschaftlichen Kalender entspricht |
| <a href="#"><u>OBJ_RECTANGLE_LABEL</u></a>   |  | Objekt "Rechteckige Marke" um eigene grafische Benutzeroberfläche zu entwerfen.   |

## OBJ\_VLINE

Vertikale Linie.



### Hinweis

Wenn Sie eine vertikale Linie erstellen, können Sie den Anzeigemodus für Linien in alle Chart-Fenster (Eigenschaft [OBJPROP\\_RAY](#)) angeben.

### Beispiel

Das folgende Skript erstellt und bewegt auf dem Chart eine vertikale Linie. Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Das Skript erstellt ein graphisches Objekt \"Vertikale Linie\".
#property description "Datum des Ankerpunkts wird als Prozentsatz der Breite"
#property description "des Chartfensters in Bars."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="VLine";      // Liniennamen
input int         InpDate=25;           // Datum der Linie %
input color       InpColor=clrRed;      // Linienfarbe
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // Linienstil
input int         InpWidth=3;           // Linienbreite
input bool        InpBack=false;        // Die Linie im Hintergrund
input bool        InpSelection=true;    // Wählen zu bewegen
input bool        InpRay=true;          // Fortsetzung der Linie nach unten
```

```

input bool      InpHidden=true;      // Ausgeblendet in der Objektliste
input long      InpZOrder=0;         // Priorität auf Mausklick
//+-----+
//| Erstellt eine vertikale Linie |
//+-----+
bool VLineCreate(const long      chart_ID=0,      // ID des Charts
                  const string   name="VLine",    // Liniennamen
                  const int      sub_window=0,    // Nummer des Unterfensters
                  datetime       time=0,          // Linienzeit
                  const color     clr=clrRed,      // Linienfarbe
                  const ENUM_LINE_STYLE style=STYLE_SOLID, // Linienstil
                  const int      width=1,         // Linienbreite
                  const bool      back=false,     // Im Hintergrund
                  const bool      selection=true,  // Wählen um zu bewegen
                  const bool      ray=true,       // Fortsetzung der Linie nach
                  const bool      hidden=true,    // Ausgeblendet in der Objektliste
                  const long      z_order=0)      // Priorität auf Mausklick
{
//--- wenn die Zeit der Linie nicht angegeben ist, wird sie auf den letzten Balken gelegt
    if(!time)
        time=TimeCurrent();
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- eine vertikale Linie erstellen
    if(!ObjectCreate(chart_ID,name,OBJ_VLINE,sub_window,time,0))
    {
        Print(__FUNCTION__,
              ": Vertikale Linie konnte nicht erstellt werden! Fehlercode = ",GetLastError());
        return(false);
    }
//--- Farbe der Linie setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- Linienstil setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- Linienbreite setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- Aktivieren (true) oder deaktivieren (false) Mausbewegung Modus
//--- Wenn Sie ein grafisches Objekt durch Funktion ObjectCreate erstellen, das Objekt
//--- kann nicht ausgewählt und verschoben werden. Innerhalb dieser Methode ist der Status
//--- true, was Sie das Objekt auswählen und verschieben erlaubt
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- aktivieren (true) oder deaktivieren (false) Anzeige der Linie in Unterfenster des Charts
    ObjectSetInteger(chart_ID,name,OBJPROP_RAY,ray);
//--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der Objektliste
    ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- setzen die Priorität für eine Mausklick-Ereignisse auf dem Chart

```

```

    ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Vertikale Linie bewegen |
//+-----+
bool VLineMove(const long   chart_ID=0,    // ID des Charts
               const string name="VLine",  // Liniennamen
               datetime     time=0)        // Linienzeit
{
//--- wenn die Zeit der Linie nicht angegeben ist, bewegen wir sie auf den letzten Balken
    if(!time)
        time=TimeCurrent();
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- eine vertikale Linie bewegen
    if(!ObjectMove(chart_ID,name,0,time,0))
    {
        Print(__FUNCTION__,
              ": Vertikale Linie konnte nicht bewegt werden! Fehlercode = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Löscht eine vertikale Linie |
//+-----+
bool VLineDelete(const long   chart_ID=0,    // ID des Charts
                 const string name="VLine")  // Liniennamen
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- eine vertikale Linie löschen
    if(!ObjectDelete(chart_ID,name))
    {
        Print(__FUNCTION__,
              ": Vertikale Linie konnte nicht gelöscht werden! Fehlercode = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{

```

```

//--- Überprüfen die Richtigkeit der Eingabeparameter
if(InpDate<0 || InpDate>100)
{
    Print("Fehler! Ungültige Eingabeparameter! ");
    return;
};
//--- Die Anzahl der sichtbaren Bars im Chart-Fenster
int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- Array speichert Datumswerte, die verwendet werden,
//--- um die Koordinaten der Ankerpunkte der Linie zu setzen und zu ändern
datetime date[];
//--- Speicher reservieren
ArrayResize(date,bars);
//--- Füllen die Anordnung von Daten
ResetLastError();
if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
{
    Print("Kann nicht den Wert der Zeit kopieren! Fehlercode = ",GetLastError());
    return;
};
//--- Wir definieren Punkte, um die Linie zu zeichnen
int d=InpDate*(bars-1)/100;
//--- eine vertikale Linie erstellen
if(!VLineCreate(0,InpName,0,date[d],InpColor,InpStyle,InpWidth,InpBack,
    InpSelection,InpRay,InpHidden,InpZOrder))
    return;
//--- den Chart neu zeichnen und 1 Sekunde warten
ChartRedraw();
Sleep(1000);
//--- jetzt bewegen wir die Linie
//--- Schleifenzähler
int h_steps=bars/2;
//--- Linie bewegen
for(int i=0;i<h_steps;i++)
{
    //--- den nächsten Wert nehmen
    if(d<bars-1)
        d+=1;
    //--- den Punkt bewegen
    if(!VLineMove(0,InpName,date[d]))
        return;
}
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
if(IsStopped())
    return;
//--- den Chart neu zeichnen
ChartRedraw();
// Verzögerung 0.03 Sekunden
Sleep(30);
};

```



```
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//--- den Kanal aus dem Chart löschen
    VLineDelete(0, InpName);
    ChartRedraw();
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//---
; }
```

## OBJ\_HLINE

Horizontale Linie.



### Beispiel

Das folgende Skript erstellt und bewegt auf dem Chart eine horizontale Linie. Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Das Skript erstellt ein graphisches Objekt \"Horizontale Linie\"
#property description "Preis des Ankerpunkts wird als Prozentsatz"
#property description "Chartfenster."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="HLine";      // Linienname
input int         InpPrice=25;           // Linienpreis in %
input color       InpColor=clrRed;       // Linienfarbe
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // Linienstil
input int         InpWidth=3;            // Linienbreite
input bool        InpBack=false;         // Die Linie im Hintergrund
input bool        InpSelection=true;     // Wählen zu bewegen
input bool        InpHidden=true;        // Ausgeblendet in der Objektliste
input long        InpZOrder=0;           // Priorität auf Mausklick
//+-----+
//| Erstellt eine horizontale Linie |
//+-----+
```

```

bool HLineCreate(const long      chart_ID=0,      // ID des Charts
                 const string   name="HLine",    // Liniennamen
                 const int      sub_window=0,    // Nummer des Unterfensters
                 double          price=0,         // Linienpreis
                 const color     clr=clrRed,      // Linienfarbe
                 const ENUM_LINE_STYLE style=STYLE_SOLID, // Linienstil
                 const int      width=1,         // Linienbreite
                 const bool      back=false,     // Im Hintergrund
                 const bool      selection=true,  // Wählen um zu bewegen
                 const bool      hidden=true,    // Ausgeblendet in der Objek
                 const long      z_order=0)       // Priorität auf Mausklick
{
//--- wenn der Preis nicht angegeben ist, setzen wir ihn gleich dem aktuellen Bid-Preis
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- eine horizontale Linie erstellen
    if(!ObjectCreate(chart_ID,name,OBJ_HLINE,sub_window,0,price))
    {
        Print(__FUNCTION__,
              ": Horizontale Linie konnte nicht erstellt werden! Fehlercode = ",GetLastError());
        return(false);
    }
//--- Farbe der Linie setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- Linienstil setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- Linienbreite setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- Aktivieren (true) oder deaktivieren (false) Mausbewegung Modus
//--- Wenn Sie ein grafisches Objekt durch Funktion ObjectCreate erstellen, das Objekt
//--- kann nicht ausgewählt und verschoben werden. Innerhalb dieser Methode ist der Status
//--- true, was Sie das Objekt auswählen und verschieben erlaubt
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der
    ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- setzen die Priorität für eine Mausklick-Ereignisse auf dem Chart
    ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Horizontale Linie bewegen |
//+-----+
bool HLineMove(const long      chart_ID=0,      // ID des Charts

```

```

        const string name="HLine", // Liniennamen
        double price=0) // Linienfarbe
    {
//--- wenn der Preis der Linie nicht angegeben ist, bewegen wir die Linie auf das Niveau
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- eine horizontale Linie bewegen
    if(!ObjectMove(chart_ID,name,0,0,price))
    {
        Print(__FUNCTION__,
            ": Horizontale Linie konnte nicht bewegt werden! Fehlercode = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
};

//+-----+
//| Löscht eine horizontale Linie |
//+-----+
bool HLineDelete(const long chart_ID=0, // ID des Charts
    const string name="HLine") // Liniennamen
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- eine horizontale Linie löschen
    if(!ObjectDelete(chart_ID,name))
    {
        Print(__FUNCTION__,
            ": Horizontale Linie konnte nicht gelöscht werden! Fehlercode = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
};

//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//--- Überprüfen die Richtigkeit der Eingabeparameter
    if(InpPrice<0 || InpPrice>100)
    {
        Print("Fehler! Ungültige Eingabeparameter! ");
        return;
    }
//--- Größe des Arrays price
    int accuracy=1000;

```

```

//--- Array speichert Preiswerte, die verwendet werden,
//--- um die Koordinaten der Ankerpunkten der Linie zu setzen und zu ändern
    double price[];
//--- Speicher reservieren
    ArrayResize(price,accuracy);
//--- Füllen den Array der Preise
//--- den Minimal- und Maximalwert der Charts finden
    double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
    double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- Schritte der Preisänderung finden und das Array füllen
    double step=(max_price-min_price)/accuracy;
    for(int i=0;i<accuracy;i++)
        price[i]=min_price+i*step;
//--- Wir definieren Punkte, um die Linie zu zeichnen
    int p=InpPrice*(accuracy-1)/100;
//--- eine horizontale Linie erstellen
    if(!HLineCreate(0,InpName,0,price[p],InpColor,InpStyle,InpWidth,InpBack,
        InpSelection,InpHidden,InpZOrder))
    {
        return;
    };
//--- den Chart neu zeichnen und 1 Sekunde warten
    ChartRedraw();
    Sleep(1000);
//--- jetzt bewegen wir die Linie
//--- Schleifenzähler
    int v_steps=accuracy/2;
//--- Linie bewegen
    for(int i=0;i<v_steps;i++)
    {
        //--- den nächsten Wert nehmen
        if(p<accuracy-1)
            p+=1;
        //--- den Punkt bewegen
        if(!HLineMove(0,InpName,price[p]))
            return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
        if(IsStopped())
            return;
        //--- den Chart neu zeichnen
        ChartRedraw();
    };
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//--- die Linie aus dem Chart löschen
    HLineDelete(0,InpName);
    ChartRedraw();
//--- 1 Sekunde Verzögerung
    Sleep(1000);

```

```
//---  
; }
```

## OBJ\_TREND

Trendline.



### Hinweis

Für die Trendlinie können Sie die weitere Anzeige nach rechts und/oder nach links (Eigenschaften [OBJPROP\\_RAY\\_RIGHT](#) und [OBJPROP\\_RAY\\_LEFT](#) jeweils) angeben.

### Beispiel

Das folgende Skript erstellt und bewegt auf dem Chart eine Trendlinie. Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Das Skript erstellt ein graphisches Objekt \"Trendlinie\"."
#property description "Koordinaten des Bezugspunktes werden in Prozent von"
#property description "der Größe des Chartfensters eingegeben."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="Trend";      // Liniennamen
input int         InpDate1=35;          // Datum des ersten Punktes in %
input int         InpPrice1=60;         // Preis des ersten Punktes in %
input int         InpDate2=65;          // Datum des zweiten Punktes in %
input int         InpPrice2=40;         // Preis des zweiten Punktes in %
input color       InpColor=clrRed;      // Linienfarbe
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // Linienstil
```

```

input int      InpWidth=2;          // Linienbreite
input bool     InpBack=false;       // Die Linie im Hintergrund
input bool     InpSelection=true;   // Wählen zu bewegen
input bool     InpRayLeft=false;    // Fortsetzung der Linie nach links
input bool     InpRayRight=false;   // Fortsetzung der Linie nach rechts
input bool     InpHidden=true;      // Ausgeblendet in der Objektliste
input long     InpZOrder=0;         // Priorität auf Mausklick
//+-----+
//| Erstellt eine Trendlinie auf angegebenen Koordinaten |
//+-----+
bool TrendCreate(const long      chart_ID=0,          // ID des Charts
                 const string    name="TrendLine",    // Liniennamen
                 const int       sub_window=0,        // Nummer des Unterfensters
                 datetime        time1=0,             // Zeit des ersten Punktes
                 double           price1=0,           // Preis des ersten Punktes
                 datetime        time2=0,             // Zeit des zweiten Punktes
                 double           price2=0,           // Preis des zweiten Punktes
                 const color      clr=clrRed,         // Linienfarbe
                 const ENUM_LINE_STYLE style=STYLE_SOLID, // Linienstil
                 const int       width=1,            // Linienbreite
                 const bool       back=false,        // Im Hintergrund
                 const bool       selection=true,     // Wählen um zu bewegen
                 const bool       ray_left=false,     // Fortsetzung der Linie nach links
                 const bool       ray_right=false,    // Fortsetzung der Linie nach rechts
                 const bool       hidden=true,        // Ausgeblendet in der Objektliste
                 const long       z_order=0)          // Priorität auf Mausklick
{
    //--- Die Koordinaten der Bezugspunkte angeben, wenn sie nicht gesetzt sind
    ChangeTrendEmptyPoints(time1,price1,time2,price2);
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- eine Trendlinien auf angegebenen Koordinaten erstellen
    if(!ObjectCreate(chart_ID,name,OBJ_TREND,sub_window,time1,price1,time2,price2))
    {
        Print(__FUNCTION__,
              ": Trendlinie konnte nicht erstellt werden! Fehlercode = ",GetLastError());
        return(false);
    }
    ;}
    //--- Farbe der Linie setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
    //--- Linienstil setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
    //--- Linienbreite setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
    //--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
    //--- Aktivieren (true) oder deaktivieren (false) Mausbewegung Modus
    //--- Wenn Sie ein grafisches Objekt durch Funktion ObjectCreate erstellen, das Objekt
    //--- kann nicht ausgewählt und verschoben werden. Innerhalb dieser Methode ist der St

```



```

//--- true, was Sie das Objekt auswählen und verschieben erlaubt
ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- aktivieren (true) oder deaktivieren (false) Fortsetzung der Linie nach links
ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- aktivieren (true) oder deaktivieren (false) Fortsetzung der Linie nach rechts
ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der
ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- setzen die Priorität für eine Mausklick-Ereignisse auf dem Chart
ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
return(true);
; }

//+-----+
//| Verschiebt den Ankerpunkt von Trendlinien |
//+-----+
bool TrendPointChange(const long   chart_ID=0,      // ID des Charts
                      const string name="TrendLine", // Liniennamen
                      const int    point_index=0,   // Nummer des Ankerpunktes
                      datetime      time=0,         // Zeitkoordinate des Punktes
                      double         price=0)        // Preiskoordinate des Punktes
{
//--- Wenn die Koordinaten des Punktes nicht angegeben sind, dann verschieben wir es
if(!time)
    time=TimeCurrent();
if(!price)
    price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- Setzen den Wert des Fehlers zurück
ResetLastError();
//--- den Ankerpunkt der Trendlinie bewegen
if(!ObjectMove(chart_ID,name,point_index,time,price))
{
    Print(__FUNCTION__,
          ": der Bezugspunkt konnte nicht bewegt werden! Fehlercode = ",GetLastError());
    return(false);
}
//--- die erfolgreiche Umsetzung
return(true);
; }

//+-----+
//| Die Funktion löscht die Trendlinie aus dem Chart. |
//+-----+
bool TrendDelete(const long   chart_ID=0,      // ID des Charts
                 const string name="TrendLine") // Liniennamen
{
//--- Setzen den Wert des Fehlers zurück
ResetLastError();
//--- eine Trendlinie löschen

```

```

    if(!ObjectDelete(chart_ID,name))
    {
        Print(__FUNCTION__,
              ": Trendlinie konnte nicht gelöscht werden! Fehlercode = ",GetLastError())
        return(false);
    }
    ;}
//--- die erfolgreiche Umsetzung
    return(true);
    ;}
//+-----+
//| Überprüft die Werte der Ankerpunkten von Trendlinie |
//| und setzt Standardwerte für leere Werte |
//+-----+
void ChangeTrendEmptyPoints(datetime &time1,double &price1,
                             datetime &time2,double &price2)
{
    //--- Wenn Zeit des ersten Punktes nicht angegeben ist, wird es auf dem aktuellen Bar
    if(!time1)
        time1=TimeCurrent();
    //--- Wenn der Preis des ersten Punktes nicht angegeben wird, dann wird es einen Wert
    if(!price1)
        price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
    //--- wenn die Zeit des zweiten Punktes nicht angegeben ist, wird sie auf 9 Baren nach
    if(!time2)
    {
        //--- Array um Eröffnungszeit der letzten 10 Bars aufzunehmen
        datetime temp[10];
        CopyTime(Symbol(),Period(),time1,10,temp);
        //--- den zweiten Punkt 9 Bars nach links vom ersten Punkt setzen
        time2=temp[0];
    }
    ;}
    //--- Wenn der Preis des zweiten Punktes nicht angegeben wird, dann wird es der Preis
    if(!price2)
        price2=price1;
    ;}
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    //--- Überprüfen die Richtigkeit der Eingabeparameter
    if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
       InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
    {
        Print("Fehler! Ungültige Eingabeparameter! ");
        return;
    }
    ;}
    //--- Die Anzahl der sichtbaren Bars im Chart-Fenster
    int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);

```

```

//--- Größe des Arrays price
    int accuracy=1000;
//--- Arrays für Werte von Daten und Preise, die für Setzung und Änderung
//--- der Koordinaten von Bezugspunkte der Linie verwendet werden
    datetime date[];
    double price[];
//--- Speicher reservieren
    ArrayResize(date,bars);
    ArrayResize(price,accuracy);
//--- Füllen die Anordnung von Daten
    ResetLastError();
    if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
    {
        Print("Kann nicht den Wert der Zeit kopieren! Fehlercode = ",GetLastError());
        return;
    }
//--- Füllen den Array der Preise
//--- den Minimal- und Maximalwert der Charts finden
    double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
    double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- Schritte der Preisänderung finden und das Array füllen
    double step=(max_price-min_price)/accuracy;
    for(int i=0;i<accuracy;i++)
        price[i]=min_price+i*step;
//--- Wir definieren Punkte, um die Linie zu zeichnen
    int d1=InpDate1*(bars-1)/100;
    int d2=InpDate2*(bars-1)/100;
    int p1=InpPrice1*(accuracy-1)/100;
    int p2=InpPrice2*(accuracy-1)/100;
//--- eine Trendlinie erstellen
    if(!TrendCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor,InpStyle,
        InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder))
    {
        return;
    }
//--- den Chart neu zeichnen und 1 Sekunde warten
    ChartRedraw();
    Sleep(1000);
//--- Jetzt werden wir die Bezugspunkte der Linie bewegen
//--- Schleifenzähler
    int v_steps=accuracy/5;
//--- Wir bewegen den ersten Bezugspunkt vertikal
    for(int i=0;i<v_steps;i++)
    {
        //--- den nächsten Wert nehmen
        if(p1>1)
            p1-=1;
        //--- den Punkt bewegen
        if(!TrendPointChange(0,InpName,0,date[d1],price[p1]))

```

```

        return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
    if(IsStopped())
        return;
//--- den Chart neu zeichnen
    ChartRedraw();
; }
//--- Wir bewegen den zweiten Bezugspunkt vertikal
for(int i=0;i<v_steps;i++)
{
    //--- den nächsten Wert nehmen
    if(p2<accuracy-1)
        p2+=1;
//--- den Punkt bewegen
    if(!TrendPointChange(0, InpName, 1, date[d2], price[p2]))
        return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
    if(IsStopped())
        return;
//--- den Chart neu zeichnen
    ChartRedraw();
; }
//--- Halb-Sekunden-Verzögerung
    Sleep(500);
//--- Schleifenzähler
    int h_steps=bars/2;
//--- Wir bewegen die beiden Bezugspunkte horizontal gleichzeitig
for(int i=0;i<h_steps;i++)
{
    //--- die nächsten Werte nehmen
    if(d1<bars-1)
        d1+=1;
    if(d2>1)
        d2-=1;
//--- Die Punkte bewegen
    if(!TrendPointChange(0, InpName, 0, date[d1], price[p1]))
        return;
    if(!TrendPointChange(0, InpName, 1, date[d2], price[p2]))
        return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
    if(IsStopped())
        return;
//--- den Chart neu zeichnen
    ChartRedraw();
// Verzögerung 0.03 Sekunden
    Sleep(30);
; }
//--- 1 Sekunde Verzögerung
    Sleep(1000);

```

```
//--- die Trendlinie löschen  
TrendDelete(0, InpName);  
ChartRedraw();  
//--- 1 Sekunde Verzögerung  
Sleep(1000);  
//---  
; }
```

## OBJ\_TRENDBYANGLE

Winkel Trendlinie.



### Hinweis

Für die Trendlinie können Sie die weitere Anzeige nach rechts und/oder nach links (Eigenschaften [OBJPROP\\_RAY\\_RIGHT](#) und [OBJPROP\\_RAY\\_LEFT](#) jeweils) angeben.

Um die Linienneigung anzugeben, können Sie den Winkel und Koordinaten des zweiten Ankerpunktes verwenden.

### Beispiel

Das folgende Skript erstellt und bewegt auf dem Chart eine Trendlinie. Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Das Skript erstellt ein graphisches Objekt \"Winkel Trendlinie\"
#property description "Koordinaten der Ankerpunkten werden in Pixeln angegeben"
#property description "Chartfenster."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="Trend";      // Liniename
input int         InpDate1=50;          // Datum des ersten Punktes in %
input int         InpPrice1=75;         // Preis des ersten Punktes in %
input int         InpAngle=0;           // Winkel der Linie
input color       InpColor=clrRed;      // Linienfarbe
```

```

input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // Linienstil
input int              InpWidth=2;         // Linienbreite
input bool             InpBack=false;      // Die Linie im Hintergrund
input bool             InpSelection=true;   // Wählen zu bewegen
input bool             InpRayLeft=false;    // Fortsetzung der Linie nach links
input bool             InpRayRight=true;    // Fortsetzung der Linie nach rechts
input bool             InpHidden=true;     // Ausgeblendet in der Objektliste
input long             InpZOrder=0;        // Priorität auf Mausklick

//+-----+
//| Erstellt eine Winkel-Trendlinie |
//+-----+

bool TrendByAngleCreate(const long      chart_ID=0,      // ID des Charts
                        const string    name="TrendLine", // Liniennamen
                        const int       sub_window=0,    // Nummer des Unterfensters
                        datetime        time=0,          // Zeit des Punktes
                        double           price=0,         // Preis des Punktes
                        const double    angle=45.0,      // Winkel
                        const color      clr=clrRed,     // Linienfarbe
                        const ENUM_LINE_STYLE style=STYLE_SOLID, // Linienstil
                        const int        width=1,        // Linienbreite
                        const bool        back=false,     // Im Hintergrund
                        const bool        selection=true,  // Wählen um zu bewegen
                        const bool        ray_left=false, // Fortsetzung der Linie nach links
                        const bool        ray_right=true, // Fortsetzung der Linie nach rechts
                        const bool        hidden=true,    // Ausgeblendet in der Objektliste
                        const long        z_order=0)      // Priorität auf Mausklick
{
    //--- Um es bequem wäre, die Trendlinie mit dem Maus zu bewegen, erstellen wir einen z
    datetime time2=0;
    double   price2=0;

    //--- Die Koordinaten der Bezugspunkte angeben, wenn sie nicht gesetzt sind
    ChangeTrendEmptyPoints(time,price,time2,price2);

    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();

    //--- eine Trendlinie auf zwei Punkte erstellen
    if(!ObjectCreate(chart_ID,name,OBJ_TRENDBYANGLE,sub_window,time,price,time2,price2))
    {
        Print(__FUNCTION__,
              ": Trendlinie konnte nicht erstellt werden! Fehlercode = ",GetLastError());
        return(false);
    }
    ;}

    //--- Den Winkel der Trendlinie ändern; und in den Prozess der Veränderung des Winkels
    //--- Punktes der Linie automatisch neu definiert in Übereinstimmung mit dem neuen Winkel
    ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);

    //--- Farbe der Linie setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);

    //--- Linienstil setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);

    //--- Linienbreite setzen

```

```

    ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- Aktivieren (true) oder deaktivieren (false) Mausbewegung Modus
//--- Wenn Sie ein grafisches Objekt durch Funktion ObjectCreate erstellen, das Objekt
//--- kann nicht ausgewählt und verschoben werden. Innerhalb dieser Methode ist der St
//--- true, was Sie das Objekt auswählen und verschieben erlaubt
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- aktivieren (true) oder deaktivieren (false) Fortsetzung der Linie nach links
    ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- aktivieren (true) oder deaktivieren (false) Fortsetzung der Linie nach rechts
    ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der
    ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- setzen die Priorität für eine Mausklick-Ereignisse auf dem Chart
    ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Ändert die Koordinaten der Ankerpunkte der Trendlinie |
//+-----+
bool TrendPointChange(const long   chart_ID=0,      // ID des Charts
                     const string name="TrendLine", // Liniennamen
                     datetime      time=0,          // Zeitkoordinate des Punktes
                     double         price=0)         // Preiskoordinate des Punktes
{
//--- Wenn die Koordinaten des Punktes nicht angegeben sind, dann verschieben wir es
    if(!time)
        time=TimeCurrent();
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- den Ankerpunkt der Trendlinie bewegen
    if(!ObjectMove(chart_ID,name,0,time,price))
    {
        Print(__FUNCTION__,
              ": der Bezugspunkt konnte nicht bewegt werden! Fehlercode = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Ändert den Winkel der Trendlinie |
//+-----+
bool TrendAngleChange(const long   chart_ID=0,      // ID des Charts

```



```

        const string name="TrendLine", // Name der Trendlinie
        const double angle=45)        // Winkel der Trendlinie
    {
//--- Setzen den Wert des Fehlers zurück
        ResetLastError();
//--- den Winkel der Trendlinie ändern
        if(!ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle))
        {
            Print(__FUNCTION__,
                ": Kann den Trendlinie Winkel nicht ändern! Fehlercode = ",GetLastError())
            return(false);
        };
//--- die erfolgreiche Umsetzung
        return(true);
    };
//+-----+
//| Löscht die Trendlinie |
//+-----+
bool TrendDelete(const long   chart_ID=0,      // ID des Charts
                 const string name="TrendLine") // Liniename
{
//--- Setzen den Wert des Fehlers zurück
        ResetLastError();
//--- eine Trendlinie löschen
        if(!ObjectDelete(chart_ID,name))
        {
            Print(__FUNCTION__,
                ": Trendlinie konnte nicht gelöscht werden! Fehlercode = ",GetLastError())
            return(false);
        };
//--- die erfolgreiche Umsetzung
        return(true);
    };
//+-----+
//| Überprüft die Werte der Ankerpunkten von Trendlinie |
//| und setzt Standardwerte für leere Werte |
//+-----+
void ChangeTrendEmptyPoints(datetime &time1,double &price1,
                           datetime &time2,double &price2)
{
//--- Wenn Zeit des ersten Punktes nicht angegeben ist, wird es auf dem aktuellen Bar
        if(!time1)
            time1=TimeCurrent();
//--- Wenn der Preis des ersten Punktes nicht angegeben wird, dann wird es einen Wert
        if(!price1)
            price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- setzen wir die Koordinaten des zweiten Hilfspunktes
//--- der zweite Punkt liegt 9 Balken nach links und hat den gleichen Preis
        datetime second_point_time[10];

```

```

CopyTime(Symbol(),Period(),time1,10,second_point_time);
time2=second_point_time[0];
price2=price1;
;}
//+-----+
/// Script program start function |
//+-----+
void OnStart()
{
//--- Überprüfen die Richtigkeit der Eingabeparameter
if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100)
{
Print("Fehler! Ungültige Eingabeparameter! ");
return;
;}
//--- Die Anzahl der sichtbaren Bars im Chart-Fenster
int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- Größe des Arrays price
int accuracy=1000;
//--- Arrays für Werte von Daten und Preise, die für Setzung und Änderung
//--- der Koordinaten von Bezugspunkte der Linie verwendet werden
datetime date[];
double price[];
//--- Speicher reservieren
ArrayResize(date,bars);
ArrayResize(price,accuracy);
//--- Füllen die Anordnung von Daten
ResetLastError();
if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
{
Print("Kann nicht den Wert der Zeit kopieren! Fehlercode = ",GetLastError());
return;
;}
//--- Füllen den Array der Preise
//--- den Minimal- und Maximalwert der Charts finden
double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- Schritte der Preisänderung finden und das Array füllen
double step=(max_price-min_price)/accuracy;
for(int i=0;i<accuracy;i++)
price[i]=min_price+i*step;
//--- Wir definieren Punkte, um die Linie zu zeichnen
int d1=InpDate1*(bars-1)/100;
int p1=InpPrice1*(accuracy-1)/100;
//--- eine Trendlinie erstellen
if(!TrendByAngleCreate(0,InpName,0,date[d1],price[p1],InpAngle,InpColor,InpStyle,
InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder))
{
return;
}

```

```

    ;}
//--- den Chart neu zeichnen und 1 Sekunde warten
    ChartRedraw();
    Sleep(1000);
//--- jetzt bewegen und drehen wir die Linie
//--- Schleifenzähler
    int v_steps=accuracy/2;
//--- bewegen wir den Ankerpunkt und ändern wir den Winkel der Linie
    for(int i=0;i<v_steps;i++)
    {
        //--- den nächsten Wert nehmen
        if(p1>1)
            p1-=1;
        //--- den Punkt bewegen
        if(!TrendPointChange(0,InpName,date[d1],price[p1]))
            return;
        if(!TrendAngleChange(0,InpName,18*(i+1)))
            return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
        if(IsStopped())
            return;
        //--- den Chart neu zeichnen
        ChartRedraw();
    }
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//--- die Linie aus dem Chart löschen
    TrendDelete(0,InpName);
    ChartRedraw();
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//---
    ;}

```

## OBJ\_CYCLES

Zyklische Linien.



### Hinweis

Der Abstand zwischen den Linien wird durch die Zeitkoordinaten der zwei Ankerpunkte des Objekts angegeben.

### Beispiel

Das folgende Skript erstellt und bewegt auf dem Chart zyklische Linien. Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Das Skript erstellt zyklische Linien auf dem Chart."
#property description "Koordinaten der Ankerpunkten werden in Prozente angegeben"
#property description "von der Größe des Chartfensters eingegeben."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="Cycles";    // Objektname
input int         InpDate1=10;         // Datum des ersten Punktes in %
input int         InpPrice1=45;        // Preis des ersten Punktes in %
input int         InpDate2=20;        // Datum des zweiten Punktes in %
input int         InpPrice2=55;        // Preis des zweiten Punktes in %
input color       InpColor=clrRed;     // Farbe der zyklischen Linien
input ENUM_LINE_STYLE InpStyle=STYLE_DOT; // Stil der zyklischen Linien
```

```

input int      InpWidth=1;          // Breite der zyklischen Linien
input bool     InpBack=false;       // Objekt im Hintergrund
input bool     InpSelection=true;   // Wählen um zu bewegen
input bool     InpHidden=true;     // Ausgeblendet in der Objektliste
input long     InpZOrder=0;        // Priorität auf Mausklick

//+-----+
//| Erstellt Zyklische Linien                                     |
//+-----+

bool CyclesCreate(const long      chart_ID=0,          // ID des Charts
                  const string   name="Cycles",       // Objektname
                  const int      sub_window=0,        // Nummer des Unterfensters
                  datetime       time1=0,             // Zeit des ersten Punktes
                  double         price1=0,            // Preis des ersten Punktes
                  datetime       time2=0,             // Zeit des zweiten Punktes
                  double         price2=0,            // Preis des zweiten Punktes
                  const color     clr=clrRed,         // Farbe der zyklischen Linien
                  const ENUM_LINE_STYLE style=STYLE_SOLID, // Stil der zyklischen Linien
                  const int      width=1,            // Breite der zyklischen Linien
                  const bool     back=false,         // Im Hintergrund
                  const bool     selection=true,     // Wählen um zu bewegen
                  const bool     hidden=true,        // Ausgeblendet in der Liste
                  const long      z_order=0)         // Priorität auf Mausklick
{
    //--- Die Koordinaten der Bezugspunkte angeben, wenn sie nicht gesetzt sind
    ChangeCyclesEmptyPoints(time1,price1,time2,price2);
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- zyklische Linien auf angegebenen Koordinaten erstellen
    if(!ObjectCreate(chart_ID,name,OBJ_CYCLES,sub_window,time1,price1,time2,price2))
    {
        Print(__FUNCTION__,
              ": Zyklische Linien konnten nicht erstellt werden! Fehlercode = ",GetLastError());
        return(false);
    }
    //--- Linienfarbe angeben
    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
    //--- Linienstil angeben
    ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
    //--- Linienbreite angeben
    ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
    //--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
    //--- aktivieren (true) oder deaktivieren (false) Bewegung der Linien mit dem Maus
    //--- Wenn Sie ein grafisches Objekt durch Funktion ObjectCreate erstellen, das Objekt
    //--- kann nicht ausgewählt und verschoben werden. Innerhalb dieser Methode ist der Stil
    //--- true, was Sie das Objekt auswählen und verschieben erlaubt
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
    //--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der

```

```

    ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- setzen die Priorität für eine Mausklick-Ereignisse auf dem Chart
    ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
    return(true);
; }
//+-----+
//| Bewegt den Bezugspunkt |
//+-----+
bool CyclesPointChange(const long   chart_ID=0,    // ID des Charts
                      const string name="Cycles", // Objektname
                      const int    point_index=0, // Nummer des Punktes
                      datetime      time=0,       // Zeitkoordinate des Punktes
                      double         price=0)      // Preiskoordinate des Punktes
{
//--- Wenn die Koordinaten des Punktes nicht angegeben sind, dann verschieben wir es a
    if(!time)
        time=TimeCurrent();
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- bewegen den Bezugspunkt
    if(!ObjectMove(chart_ID,name,point_index,time,price))
    {
        Print(__FUNCTION__,
              ": der Bezugspunkt konnte nicht bewegt werden! Fehlercode = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
; }
//+-----+
//| Löscht Zyklische Linien |
//+-----+
bool CyclesDelete(const long   chart_ID=0,    // ID des Charts
                  const string name="Cycles") // Objektname
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Zyklische Linien löschen
    if(!ObjectDelete(chart_ID,name))
    {
        Print(__FUNCTION__,
              ": Zyklische Linien konnten nicht gelöscht werden! Fehlercode = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);

```

```

; }
//+-----+
//| Überprüft die Werte der Ankerpunkten von zyklische Linien |
//| und setzt Standardwerte für leere Werte |
//+-----+
void ChangeCyclesEmptyPoints(datetime &time1,double &price1,
                             datetime &time2,double &price2)
{
//--- Wenn Zeit des ersten Punktes nicht angegeben ist, wird es auf dem aktuellen Bar
    if(!time1)
        time1=TimeCurrent();
//--- Wenn der Preis des ersten Punktes nicht angegeben wird, dann wird es einen Wert
    if(!price1)
        price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- wenn die Zeit des zweiten Punktes nicht angegeben ist, wird sie auf 9 Baren nach
    if(!time2)
    {
        //--- Array um Eröffnungszeit der letzten 10 Bars aufzunehmen
        datetime temp[10];
        CopyTime(Symbol(),Period(),time1,10,temp);
        //--- den zweiten Punkt 9 Bars nach links vom ersten Punkt setzen
        time2=temp[0];
    }
; }
//--- Wenn der Preis des zweiten Punktes nicht angegeben wird, dann wird es der Preis
    if(!price2)
        price2=price1;
; }
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//--- Überprüfen die Richtigkeit der Eingabeparameter
    if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
        InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
    {
        Print("Fehler! Ungültige Eingabeparameter! ");
        return;
    }
; }
//--- Die Anzahl der sichtbaren Bars im Chart-Fenster
    int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- Größe des Arrays price
    int accuracy=1000;
//--- Arrays für Werte von Daten und Preise, die für Setzung und Änderung
//--- der Koordinaten der Ankerpunkten der zyklische Linien verwendet werden
    datetime date[];
    double price[];
//--- Speicher reservieren
    ArrayResize(date,bars);

```

```

    ArrayResize(price,accuracy);
//--- Füllen die Anordnung von Daten
    ResetLastError();
    if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
    {
        Print("Kann nicht den Wert der Zeit kopieren! Fehlercode = ",GetLastError());
        return;
    }
//--- Füllen den Array der Preise
//--- den Minimal- und Maximalwert der Charts finden
    double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
    double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- Schritte der Preisänderung finden und das Array füllen
    double step=(max_price-min_price)/accuracy;
    for(int i=0;i<accuracy;i++)
        price[i]=min_price+i*step;
//--- Definieren die Punkte um die zyklische Linien zu zeichnen
    int d1=InpDate1*(bars-1)/100;
    int d2=InpDate2*(bars-1)/100;
    int p1=InpPrice1*(accuracy-1)/100;
    int p2=InpPrice2*(accuracy-1)/100;
//--- eine Trendlinie erstellen
    if(!CyclesCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor,
        InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
    {
        return;
    }
//--- den Chart neu zeichnen und 1 Sekunde warten
    ChartRedraw();
    Sleep(1000);
//--- jetzt bewegen wir die Ankerpunkte
//--- Schleifenzähler
    int h_steps=bars/5;
//--- den zweiten Punkt bewegen
    for(int i=0;i<h_steps;i++)
    {
        //--- den nächsten Wert nehmen
        if(d2<bars-1)
            d2+=1;
        //--- den Punkt bewegen
        if(!CyclesPointChange(0,InpName,1,date[d2],price[p2]))
            return;
    }
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
    if(IsStopped())
        return;
//--- den Chart neu zeichnen
    ChartRedraw();
// Verzögerung 0.05 Sekunden
    Sleep(50);

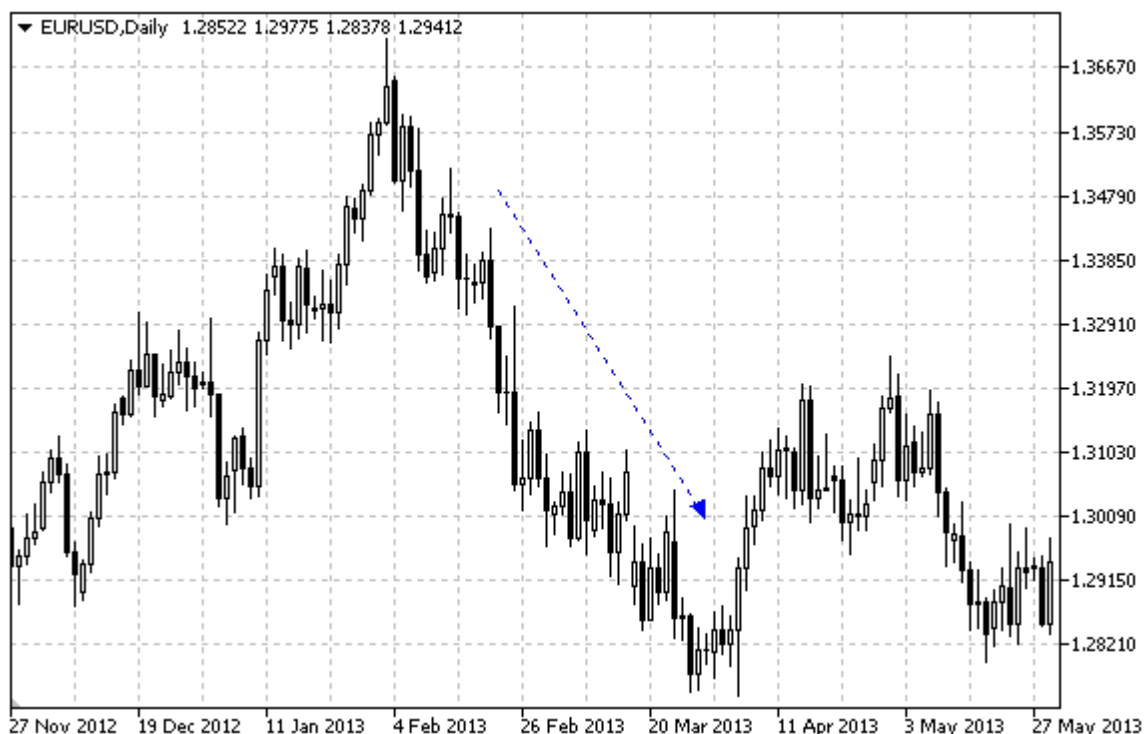
```



```
    ;}
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//--- Schleifenzähler
    h_steps=bars/4;
//--- den ersten Punkt bewegen
    for(int i=0;i<h_steps;i++)
    {
        //--- den nächsten Wert nehmen
        if(d1<bars-1)
            d1+=1;
        //--- den Punkt bewegen
        if(!CyclesPointChange(0,InpName,0,date[d1],price[p1]))
            return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
        if(IsStopped())
            return;
        //--- den Chart neu zeichnen
        ChartRedraw();
        // Verzögerung 0.05 Sekunden
        Sleep(50);
    }
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//--- das Objekt aus dem Chart löschen
    CyclesDelete(0,InpName);
    ChartRedraw();
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//---
    ;}
```

## OBJ\_ARROWED\_LINE

Linie mit einem Pfeil.



### Beispiel

Das folgende Skript erstellt und bewegt auf dem Chart eine Linie mit einem Pfeil. Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Das Skript erstellt das graphische Objekt \"Linie mit einem Pfeil\"
#property description "Koordinaten des Bezugspunktes werden in Prozent von \"
#property description "der Größe des Chartfensters eingegeben."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="ArrowedLine"; // Name der Linie
input int         InpDate1=35;           // Datum des ersten Punktes in %
input int         InpPrice1=60;           // Preis des ersten Punktes in %
input int         InpDate2=65;           // Datum des zweiten Punktes in %
input int         InpPrice2=40;           // Preis des zweiten Punktes in %
input color       InpColor=clrRed;        // Linienfarbe
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // Linienstil
input int         InpWidth=2;             // Linienbreite
input bool        InpBack=false;          // Linie im Hintergrund
input bool        InpSelection=true;       // Wählen zu bewegen
input bool        InpHidden=true;         // in der Liste der Objekten ausgeblendet
input long        InpZOrder=0;            // Priorität auf Mausklick
```

```

//+-----+
//| Erstellt eine Linie mit Pfeil auf den angegebenen Koordinaten |
//+-----+
bool ArrowedLineCreate(const long      chart_ID=0,          // ID des Charts
                      const string    name="ArrowedLine",  // Liniennamen
                      const int        sub_window=0,       // Nummer des Unterfensters
                      datetime         time1=0,            // Zeit des ersten Punktes
                      double           price1=0,           // Preis des ersten Punktes
                      datetime         time2=0,            // Zeit des zweiten Punktes
                      double           price2=0,           // Preis des zweiten Punktes
                      const color      clr=clrRed,         // Linienfarbe
                      const ENUM_LINE_STYLE style=STYLE_SOLID, // Linienstil
                      const int        width=1,           // Linienbreite
                      const bool        back=false,       // Im Hintergrund
                      const bool        selection=true,    // Wählen zu bewegen
                      const bool        hidden=true,       // Verborgenen in der Liste
                      const long        z_order=0)         // Priorität für Mausereignisse
{
    //--- Die Koordinaten der Bezugspunkte angeben, wenn sie nicht gesetzt sind
    ChangeArrowedLineEmptyPoints(time1,price1,time2,price2);
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Wir erstellen eine Linie mit einem Pfeil an den angegebenen Koordinaten
    if(!ObjectCreate(chart_ID,name,OBJ_ARROWED_LINE,sub_window,time1,price1,time2,price2))
    {
        Print(__FUNCTION__,
              ": Eine Linie mit dem Pfeil konnte nicht erstellt werden! Fehlercode = ",GetLastError());
        return(false);
    }
    //--- Farbe der Linie setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
    //--- Linienstil setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
    //--- Linienbreite setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
    //--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
    //--- Aktivieren (true) oder deaktivieren (false) Mausbewegung Modus
    //--- Wenn Sie ein grafisches Objekt durch Funktion ObjectCreate erstellen, das Objekt
    //--- kann nicht ausgewählt und verschoben werden. Innerhalb dieser Methode ist der Status
    //--- true, was Sie das Objekt auswählen und verschieben erlaubt
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
    //--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der Liste
    ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
    //--- setzen die Priorität für eine Mausklick-Ereignisse auf dem Chart
    ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
    //--- die erfolgreiche Umsetzung
    return(true);
}

```

```

; }
//+-----+
//| Bewegt den Bezugspunkt der Linie mit einem Pfeil |
//+-----+
bool ArrowedLinePointChange(const long   chart_ID=0,           // ID des Charts
                           const string name="ArrowedLine",    // Liniennamen
                           const int    point_index=0,         // Nummer des Bezugspunktes
                           datetime     time=0,               // Zeitkoordinate des Bezugspunktes
                           double       price=0)               // Preiskoordinate des Bezugspunktes
{
//--- Wenn die Koordinaten des Punktes nicht angegeben sind, dann verschieben wir es auf den aktuellen Wert
if(!time)
    time=TimeCurrent();
if(!price)
    price=SymbolInfoDouble(Symbol(), SYMBOL_BID);
//--- Setzen den Wert des Fehlers zurück
ResetLastError();
//--- bewegen den Bezugspunkt der Linie
if(!ObjectMove(chart_ID,name,point_index,time,price))
{
    Print(__FUNCTION__,
          ": der Bezugspunkt konnte nicht bewegt werden! Fehlercode = ",GetLastError());
    return(false);
}
; }
//--- die erfolgreiche Umsetzung
return(true);
; }
//+-----+
//| Die Funktion löscht die Linie mit dem Pfeil aus dem Chart |
//+-----+
bool ArrowedLineDelete(const long   chart_ID=0,           // ID des Charts
                      const string name="ArrowedLine")    // Liniennamen
{
//--- Setzen den Wert des Fehlers zurück
ResetLastError();
//--- Die Linie mit einem Pfeil löschen
if(!ObjectDelete(chart_ID,name))
{
    Print(__FUNCTION__,
          ": Eine Linie mit dem Pfeil konnte nicht gelöscht werden! Fehlercode = ",GetLastError());
    return(false);
}
; }
//--- die erfolgreiche Umsetzung
return(true);
; }
//+-----+
//| Überprüft die Werte der Bezugspunkte der Linie für leere Werte |
//| setzt Standardwerte |
//+-----+

```

```

void ChangeArrowedLineEmptyPoints(datetime &time1,double &price1,
                                   datetime &time2,double &price2)
{
    //--- Wenn Zeit des ersten Punktes nicht angegeben ist, wird es auf dem aktuellen Bar
    if(!time1)
        time1=TimeCurrent();
    //--- Wenn der Preis des ersten Punktes nicht angegeben wird, dann wird es einen Wert
    if(!price1)
        price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
    //--- Wenn Zeit des zweiten Punktes nicht angegeben ist, wird er 9 Bars nach links von
    if(!time2)
    {
        //--- Array um Eröffnungszeit der letzten 10 Bars aufzunehmen
        datetime temp[10];
        CopyTime(Symbol(),Period(),time1,10,temp);
        //--- den zweiten Punkt 9 Bars nach links vom ersten Punkt setzen
        time2=temp[0];
    }
    //--- Wenn der Preis des zweiten Punktes nicht angegeben wird, dann wird es der Preis
    if(!price2)
        price2=price1;
    ;}

//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    //--- Überprüfen die Richtigkeit der Eingabeparameter
    if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
        InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
    {
        Print("Fehler! Ungültige Eingabeparameter! ");
        return;
    }
    ;}

    //--- Die Anzahl der sichtbaren Bars im Chart-Fenster
    int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
    //--- Größe des Arrays price
    int accuracy=1000;
    //--- Arrays für Werte von Daten und Preise, die für Setzung und Änderung
    //--- der Koordinaten von Bezugspunkte der Linie verwendet werden
    datetime date[];
    double price[];
    //--- Speicher reservieren
    ArrayResize(date,bars);
    ArrayResize(price,accuracy);
    //--- Füllen die Anordnung von Daten
    ResetLastError();
    if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
    {

```

```

        Print("Kann nicht den Wert der Zeit kopieren! Fehlercode = ",GetLastError());
        return;
    };
}

//--- Füllen den Array der Preise
//--- den Minimal- und Maximalwert der Charts finden
double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- Schritte der Preisänderung finden und das Array füllen
double step=(max_price-min_price)/accuracy;
for(int i=0;i<accuracy;i++)
    price[i]=min_price+i*step;
//--- Wir definieren Punkte, um die Linie zu zeichnen
int d1=InpDate1*(bars-1)/100;
int d2=InpDate2*(bars-1)/100;
int p1=InpPrice1*(accuracy-1)/100;
int p2=InpPrice2*(accuracy-1)/100;
//--- Die Linie mit einem Pfeil erstellen
if(!ArrowedLineCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],
    InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
{
    return;
};
//--- den Chart neu zeichnen und 1 Sekunde warten
ChartRedraw();
Sleep(1000);
//--- Jetzt werden wir die Bezugspunkte der Linie bewegen
//--- Schleifenzähler
int v_steps=accuracy/5;
//--- Wir bewegen den zweiten Bezugspunkt vertikal
for(int i=0;i<v_steps;i++)
{
    //--- den nächsten Wert nehmen
    if(p2<accuracy-1)
        p2+=1;
    //--- den Punkt bewegen
    if(!ArrowedLinePointChange(0,InpName,1,date[d2],price[p2]))
        return;
}
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
if(IsStopped())
    return;
//--- den Chart neu zeichnen
ChartRedraw();
};
//--- Wir bewegen den ersten Bezugspunkt vertikal
for(int i=0;i<v_steps;i++)
{
    //--- den nächsten Wert nehmen
    if(p1>1)
        p1-=1;

```

```

    //--- den Punkt bewegen
    if(!ArrowedLinePointChange(0, InpName, 0, date[d1], price[p1]))
        return;
    //--- Überprüfen die Fakten von Zwangsabschaltung der Skript
    if(IsStopped())
        return;
    //--- den Chart neu zeichnen
    ChartRedraw();
    ;}
//--- Halb-Sekunden-Verzögerung
Sleep(500);
//--- Schleifenzähler
int h_steps=bars/2;
//--- Wir bewegen die beiden Bezugspunkte horizontal gleichzeitig
for(int i=0; i<h_steps; i++)
{
    //--- die nächsten Werte nehmen
    if(d1<bars-1)
        d1+=1;
    if(d2>1)
        d2-=1;
    //--- Die Punkte bewegen
    if(!ArrowedLinePointChange(0, InpName, 0, date[d1], price[p1]))
        return;
    if(!ArrowedLinePointChange(0, InpName, 1, date[d2], price[p2]))
        return;
    //--- Überprüfen die Fakten von Zwangsabschaltung der Skript
    if(IsStopped())
        return;
    //--- den Chart neu zeichnen
    ChartRedraw();
    // Verzögerung 0.03 Sekunden
    Sleep(30);
    ;}
//--- 1 Sekunde Verzögerung
Sleep(1000);
//--- Die Linie mit einem Pfeil löschen
ArrowedLineDelete(0, InpName);
ChartRedraw();
//--- 1 Sekunde Verzögerung
Sleep(1000);
//---
; }

```

## OBJ\_CHANNEL

Äquidistanter Kanal.



### Hinweis

Für einen äquidistanten Kanal können Sie die weitere Anzeige nach rechts und/oder nach links (Eigenschaften [OBJPROP\\_RAY\\_RIGHT](#) und [OBJPROP\\_RAY\\_LEFT](#) jeweils) angeben. Sie können auch die Farbfüllung des Kanals angeben.

### Beispiel

Das folgende Skript erstellt und bewegt auf dem Chart einen äquidistanten Kanal. Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Das Skript erstellt ein graphisches Objekt \"Äquidistanter Kanal\"
#property description "Koordinaten der Ankerpunkten werden in Pixeln angegeben"
#property description "Chartfenster."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="Channel";    // Kanalname
input int         InpDate1=25;          // Datum des ersten Punktes in %
input int         InpPrice1=60;         // Preis des ersten Punktes in %
input int         InpDate2=65;          // Datum des zweiten Punktes in %
input int         InpPrice2=80;         // Preis des zweiten Punktes in %
input int         InpDate3=30;          // Datum des dritten Punktes in %
```



```

input int          InpPrice3=40;          // Preis des dritten Punktes in %
input color        InpColor=clrRed;       // Farbe des Kanals
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // Stil der Kanallinie
input int          InpWidth=2;           // Linienbreite
input bool         InpBack=false;         // Kanal im Hintergrund
input bool         InpFill=false;         // Farbfüllung des Kanals
input bool         InpSelection=true;      // Wählen zu bewegen
input bool         InpRayLeft=false;      // Fortsetzung des Kanals nach links
input bool         InpRayRight=false;     // Fortsetzung des Kanals nach rechts
input bool         InpHidden=true;        // Ausgeblendet in der Objektliste
input long         InpZOrder=0;           // Priorität auf Mausclick

//+-----+
//| Erstellt einen äquidistanten Kanal auf angegebenen Koordinaten |
//+-----+

bool ChannelCreate(const long      chart_ID=0,          // ID des Charts
                  const string    name="Channel",       // Kanalname
                  const int       sub_window=0,         // Nummer des Unterfensters
                  datetime         time1=0,             // Zeit des ersten Punktes
                  double          price1=0,            // Preis des ersten Punktes
                  datetime         time2=0,             // Zeit des zweiten Punktes
                  double          price2=0,            // Preis des zweiten Punktes
                  datetime         time3=0,             // Zeit des dritten Punktes
                  double          price3=0,            // Preis des dritten Punktes
                  const color      clr=clrRed,          // Kanalfarbe
                  const ENUM_LINE_STYLE style=STYLE_SOLID, // Stil der Kanallinien
                  const int       width=1,             // Breite der Kanallinien
                  const bool       fill=false,         // Farbfüllung des Kanals
                  const bool       back=false,         // Im Hintergrund
                  const bool       selection=true,      // Auswählen um zu bewegen
                  const bool       ray_left=false,     // Fortsetzung des Kanals
                  const bool       ray_right=false,    // Fortsetzung des Kanals
                  const bool       hidden=true,        // Ausgeblendet in der Obj
                  const long       z_order=0)           // Priorität auf Mausclick
{
//--- Die Koordinaten der Bezugspunkte angeben, wenn sie nicht gesetzt sind
    ChangeChannelEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Kanal auf angegebenen Koordinaten erstellen
    if(!ObjectCreate(chart_ID,name,OBJ_CHANNEL,sub_window,time1,price1,time2,price2,time3,price3))
    {
        Print(__FUNCTION__,
              ": Kanal konnte nicht erstellt werden! Fehlercode = ",GetLastError());
        return(false);
    }
//--- Kanalfarbe setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- Stil der Kanallinien einstellen
    ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);

```

```

//--- Breite der Kanallinien einstellen
    ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- aktivieren (true) oder deaktivieren (false) Kanalfüllung
    ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- aktivieren (true) oder deaktivieren (false) Wählen des Kanals für Bewegung
//--- Wenn Sie ein grafisches Objekt durch Funktion ObjectCreate erstellen, das Objekt
//--- kann nicht ausgewählt und verschoben werden. Innerhalb dieser Methode ist der St
//--- true, was Sie das Objekt auswählen und verschieben erlaubt
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- aktivieren (true) oder deaktivieren (false) Fortsetzung des Kanals nach links
    ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- aktivieren (true) oder deaktivieren (false) Fortsetzung des Kanals nach rechts
    ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der
    ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- setzen die Priorität für eine Mausklick-Ereignisse auf dem Chart
    ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
    return(true);
; }

//+-----+
//| Verschiebt die Kanalbindung |
//+-----+
bool ChannelPointChange(const long   chart_ID=0,    // ID des Charts
                        const string name="Channel", // Kanalname
                        const int    point_index=0,  // Nummer des Punktes
                        datetime      time=0,        // Zeitkoordinate des Punktes
                        double         price=0)       // Preiskoordinate des Punktes
{
//--- Wenn die Koordinaten des Punktes nicht angegeben sind, dann verschieben wir es
    if(!time)
        time=TimeCurrent();
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- bewegen den Bezugspunkt
    if(!ObjectMove(chart_ID,name,point_index,time,price))
    {
        Print(__FUNCTION__,
              ": der Bezugspunkt konnte nicht bewegt werden! Fehlercode = ",GetLastError());
        return(false);
    }
; }
//--- die erfolgreiche Umsetzung
    return(true);
; }

```

```

//+-----+
//| Löscht den Kanal |
//+-----+
bool ChannelDelete(const long chart_ID=0, // ID des Charts
                  const string name="Channel") // Kanalname
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- den Kanal löschen
    if(!ObjectDelete(chart_ID,name))
    {
        Print(__FUNCTION__,
              ": kann nicht den Kanal löschen! Fehlercode = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Überprüft die Werte der Kanalbindungspunkten für leere Werte |
//| setzt Standardwerte |
//+-----+
void ChangeChannelEmptyPoints(datetime &time1,double &price1,datetime &time2,
                             double &price2,datetime &time3,double &price3)
{
//--- wenn die Zeit des zweiten (rechten) Punktes nicht angegeben ist, wird sie auf de
    if(!time2)
        time2=TimeCurrent();
//--- wenn der Preis des zweiten Punktes nicht angegeben ist, wird er dem Bid-Wert gle
    if(!price2)
        price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- wenn die Zeit des ersten (linken) Punktes nicht angegeben ist, wird sie auf 9 Ba
    if(!time1)
    {
        //--- Array um Eröffnungszeit der letzten 10 Bars aufzunehmen
        datetime temp[10];
        CopyTime(Symbol(),Period(),time2,10,temp);
        //--- stellen wir den ersten Punkt 9 Baren links dem zweiten Punkt
        time1=temp[0];
    }
//--- wenn der Preis des ersten Punktes nicht angegeben ist, bewegen wir ihn 300 Punkt
    if(!price1)
        price1=price2+300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- wenn die Zeit des dritten Punktes nicht angegeben ist, wird sie der Zeit des zw
    if(!time3)
        time3=time1;
//--- wenn der Preis des dritten Punktes nicht angegeben ist, wird er dem Preis des zw
    if(!price3)
        price3=price2;
}

```

```

; }
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//--- Überprüfen die Richtigkeit der Eingabeparameter
if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
    InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
    InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
{
    Print("Fehler! Ungültige Eingabeparameter! ");
    return;
}
; }

//--- Die Anzahl der sichtbaren Bars im Chart-Fenster
int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- Größe des Arrays price
int accuracy=1000;
//--- Arrays für Werte von Daten und Preise, die für Setzung und Änderung
//--- der Koordinaten der Kanalbindungspunkten verwendet werden
datetime date[];
double price[];
//--- Speicher reservieren
ArrayResize(date,bars);
ArrayResize(price,accuracy);
//--- Füllen die Anordnung von Daten
ResetLastError();
if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
{
    Print("Kann nicht den Wert der Zeit kopieren! Fehlercode = ",GetLastError());
    return;
}
; }

//--- Füllen den Array der Preise
//--- den Minimal- und Maximalwert der Charts finden
double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- Schritte der Preisänderung finden und das Array füllen
double step=(max_price-min_price)/accuracy;
for(int i=0;i<accuracy;i++)
    price[i]=min_price+i*step;
//--- Definieren die Punkte um den Kanal zu zeichnen
int d1=InpDate1*(bars-1)/100;
int d2=InpDate2*(bars-1)/100;
int d3=InpDate3*(bars-1)/100;
int p1=InpPrice1*(accuracy-1)/100;
int p2=InpPrice2*(accuracy-1)/100;
int p3=InpPrice3*(accuracy-1)/100;
//--- Erstellen einen äquidistanten Kanal
if(!ChannelCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],price

```

```

        InpStyle, InpWidth, InpFill, InpBack, InpSelection, InpRayLeft, InpRayRight, InpHidden,
    {
        return;
    };
}

//--- den Chart neu zeichnen und 1 Sekunde warten
ChartRedraw();
Sleep(1000);

//--- jetzt bewegen wir die Kanalbindungspunkte
//--- Schleifenzähler
int h_steps=bars/6;
//--- den zweiten Punkt bewegen
for(int i=0;i<h_steps;i++)
{
    //--- den nächsten Wert nehmen
    if(d2<bars-1)
        d2+=1;
    //--- den Punkt bewegen
    if(!ChannelPointChange(0, InpName, 1, date[d2], price[p2]))
        return;
}

//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
if(IsStopped())
    return;

//--- den Chart neu zeichnen
ChartRedraw();
// Verzögerung 0.05 Sekunden
Sleep(50);
};

//--- 1 Sekunde Verzögerung
Sleep(1000);

//--- den ersten Punkt bewegen
for(int i=0;i<h_steps;i++)
{
    //--- den nächsten Wert nehmen
    if(d1>1)
        d1-=1;
    //--- den Punkt bewegen
    if(!ChannelPointChange(0, InpName, 0, date[d1], price[p1]))
        return;
}

//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
if(IsStopped())
    return;

//--- den Chart neu zeichnen
ChartRedraw();
// Verzögerung 0.05 Sekunden
Sleep(50);
};

//--- 1 Sekunde Verzögerung
Sleep(1000);

//--- Schleifenzähler

```

```
int v_steps=accuracy/10;
//--- den dritten Punkt bewegen
for(int i=0;i<v_steps;i++)
{
    //--- den nächsten Wert nehmen
    if(p3>1)
        p3-=1;
    //--- den Punkt bewegen
    if(!ChannelPointChange(0,InpName,2,date[d3],price[p3]))
        return;
    //--- Überprüfen die Fakten von Zwangsabschaltung der Skript
    if(IsStopped())
        return;
    //--- den Chart neu zeichnen
    ChartRedraw();
};
//--- 1 Sekunde Verzögerung
Sleep(1000);
//--- den Kanal aus dem Chart löschen
ChannelDelete(0,InpName);
ChartRedraw();
//--- 1 Sekunde Verzögerung
Sleep(1000);
//---
};
```

## OBJ\_STDDEVCHANNEL

Kanal der Standardabweichung.



### Hinweis

Für das Objekt "Kanal der Standardabweichung" können Sie die weitere Anzeige nach rechts und/oder nach links (Eigenschaften [OBJPROP\\_RAY\\_RIGHT](#) und [OBJPROP\\_RAY\\_LEFT](#) jeweils) angeben. Sie können auch die Farbfüllung des Kanals angeben.

Um die Werte des Kanals der Standardabweichung zu ändern, wird Eigenschaft [OBJPROP\\_DEVIATION](#) verwendet.

### Beispiel

Das folgende Skript erstellt und bewegt auf dem Chart Objekt "Kanal der Standardabweichung". Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Das Skript erstellt ein graphisches Objekt \"Kanal der Standardabweichung\"
#property description "Koordinaten der Ankerpunkten werden in Pixeln angegeben"
#property description "Chartfenster."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="StdDevChannel";    // Kanalname
input int         InpDate1=10;                // Datum des ersten Punktes in %
input int         InpDate2=40;                // Datum des zweiten Punktes in %
input double      InpDeviation=1.0;           // Abweichung
```

```

input color      InpColor=clrRed;           // Kanalfarbe
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Stil der Kanallinien
input int        InpWidth=2;               // Kanallinienbreite
input bool       InpFill=false;           // Farbfüllung des Kanals
input bool       InpBack=false;           // Im Hintergrund
input bool       InpSelection=true;        // Wählen um zu bewegen
input bool       InpRayLeft=false;         // Fortsetzung des Kanals nach links
input bool       InpRayRight=false;        // Fortsetzung des Kanals nach rechts
input bool       InpHidden=true;          // Ausgeblendet in der Objektliste
input long       InpZOrder=0;             // Priorität auf Mausclick

//+-----+
//| Erstellt Kanal der Standardabweichung mit angegebenen Koordinaten|
//+-----+

bool StdDevChannelCreate(const long      chart_ID=0,           // ID des Charts
                        const string    name="Channel",       // Kanalname
                        const int        sub_window=0,         // Nummer des Unterf
                        datetime         time1=0,              // Zeit des ersten I
                        datetime         time2=0,              // Zeit des zweiten
                        const double     deviation=1.0,         // Abweichung
                        const color      clr=clrRed,           // Kanalfarbe
                        const ENUM_LINE_STYLE style=STYLE_SOLID, // Stil der Kanallin
                        const int        width=1,              // Breite der Kanall
                        const bool       fill=false,           // Farbfüllung des K
                        const bool       back=false,           // Im Hintergrund
                        const bool       selection=true,        // Wählen um zu bewe
                        const bool       ray_left=false,        // Fortsetzung des Kanals
                        const bool       ray_right=false,       // Fortsetzung des Kanals
                        const bool       hidden=true,           // Ausgeblendet in c
                        const long       z_order=0)             // Priorität auf Mau

{
//--- Die Koordinaten der Bezugspunkte angeben, wenn sie nicht gesetzt sind
    ChangeChannelEmptyPoints(time1,time2);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Kanal auf angegebenen Koordinaten erstellen
    if(!ObjectCreate(chart_ID,name,OBJ_STDDEVCHANNEL,sub_window,time1,0,time2,0))
    {
        Print(__FUNCTION__,
              ": Kanal der Standardabweichung konnte nicht erstellt werden! Fehlercode =
        return(false);
    }
};

//--- Wert der Abweichung setzen, Kanalbreite abhängt von ihm
    ObjectSetDouble(chart_ID,name,OBJPROP_DEVIATION,deviation);
//--- Kanalfarbe setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- Stil der Kanallinien einstellen
    ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- Breite der Kanallinien einstellen
    ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);

```



```

//--- aktivieren (true) oder deaktivieren (false) Kanalfüllung
    ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- aktivieren (true) oder deaktivieren (false) Wählen des Kanals für Bewegung
//--- Wenn Sie ein grafisches Objekt durch Funktion ObjectCreate erstellen, das Objekt
//--- kann nicht ausgewählt und verschoben werden. Innerhalb dieser Methode ist der St
//--- true, was Sie das Objekt auswählen und verschieben erlaubt
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- aktivieren (true) oder deaktivieren (false) Fortsetzung des Kanals nach links
    ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- aktivieren (true) oder deaktivieren (false) Fortsetzung des Kanals nach rechts
    ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der
    ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- setzen die Priorität für eine Mausklick-Ereignisse auf dem Chart
    ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
    return(true);
; }

//+-----+
//| Verschiebt die Kanalbindung |
//+-----+
bool StdDevChannelPointChange(const long   chart_ID=0,      // ID des Charts
                             const string name="Channel",   // Kanalname
                             const int    point_index=0,    // Nummer des Ankerpunktes
                             datetime     time=0)           // Zeitkoordinate des Anker
{
//--- wenn die Zeit des Punktes nicht angegeben ist, bewegen wir ihn auf den letzten P
    if(!time)
        time=TimeCurrent();
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- bewegen den Bezugspunkt
    if(!ObjectMove(chart_ID,name,point_index,time,0))
    {
        Print(__FUNCTION__,
              ": der Bezugspunkt konnte nicht bewegt werden! Fehlercode = ",GetLastError());
        return(false);
    }
; }
//--- die erfolgreiche Umsetzung
    return(true);
; }

//+-----+
//| Ändert Abweichung des Kanals |
//+-----+
bool StdDevChannelDeviationChange(const long   chart_ID=0,      // ID des Charts
                                  const string name="Channel",   // Kanalname

```

```

                                const double deviation=1.0) // Abweichung
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- den Winkel der Trendlinie ändern
    if(!ObjectSetDouble(chart_ID,name,OBJPROP_DEVIATION,deviation))
    {
        Print(__FUNCTION__,
              ": Konnte nicht Kanalabweichung ändern! Fehlercode = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
};

//+-----+
//| Löscht den Kanal |
//+-----+
bool StdDevChannelDelete(const long   chart_ID=0,      // ID des Charts
                        const string name="Channel") // Kanalname
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- den Kanal löschen
    if(!ObjectDelete(chart_ID,name))
    {
        Print(__FUNCTION__,
              ": kann nicht den Kanal löschen! Fehlercode = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
};

//+-----+
//| Überprüft die Werte der Kanalbindungspunkten für leere Werte |
//| setzt Standardwerte |
//+-----+
void ChangeChannelEmptyPoints(datetime &time1,datetime &time2)
{
//--- wenn die Zeit des zweiten Punktes nicht angegeben ist, wird sie auf dem aktuellen
    if(!time2)
        time2=TimeCurrent();
//--- wenn die Zeit des ersten Punktes nicht angegeben ist, wird sie auf 9 Baren nach
    if(!time1)
    {
        //--- Array um Eröffnungszeit der letzten 10 Bars aufzunehmen
        datetime temp[10];
        CopyTime(Symbol(),Period(),time2,10,temp);
        //--- stellen wir den zweiten Punkt 9 Baren links dem zweiten Punkt
        time1=temp[0];
    }
}

```

```

    ;}

    ;}

//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//--- Überprüfen die Richtigkeit der Eingabeparameter
    if(InpDate1<0 || InpDate1>100 ||
        InpDate2<0 || InpDate2>100)
    {
        Print("Fehler! Ungültige Eingabeparameter! ");
        return;
    }
    ;}

//--- Die Anzahl der sichtbaren Bars im Chart-Fenster
    int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- Größe des Arrays price
    int accuracy=1000;
//--- Arrays für Werte von Daten und Preise, die für Setzung und Änderung
//--- der Koordinaten der Kanalbindungspunkten verwendet werden
    datetime date[];
    double price[];
//--- Speicher reservieren
    ArrayResize(date,bars);
    ArrayResize(price,accuracy);
//--- Füllen die Anordnung von Daten
    ResetLastError();
    if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
    {
        Print("Kann nicht den Wert der Zeit kopieren! Fehlercode = ",GetLastError());
        return;
    }
    ;}

//--- Füllen den Array der Preise
//--- den Minimal- und Maximalwert der Charts finden
    double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
    double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- Schritte der Preisänderung finden und das Array füllen
    double step=(max_price-min_price)/accuracy;
    for(int i=0;i<accuracy;i++)
        price[i]=min_price+i*step;
//--- Definieren die Punkte um den Kanal zu zeichnen
    int d1=InpDate1*(bars-1)/100;
    int d2=InpDate2*(bars-1)/100;
//--- wir erstellen den Kanal der Standardabweichung
    if(!StdDevChannelCreate(0,InpName,0,date[d1],date[d2],InpDeviation,InpColor,InpStyle,
        InpWidth,InpFill,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder)
    {
        return;
    }
    ;}

```

```

//--- den Chart neu zeichnen und 1 Sekunde warten
    ChartRedraw();
    Sleep(1000);
//--- jetzt bewegen wir den Kanal horizontal nach rechts und erweitern ihn
//--- Schleifenzähler
    int h_steps=bars/2;
//--- Kanal bewegen
    for(int i=0;i<h_steps;i++)
    {
        //--- die nächsten Werte nehmen
        if(d1<bars-1)
            d1+=1;
        if(d2<bars-1)
            d2+=1;
        //--- Ankerpunkte bewegen
        if(!StdDevChannelPointChange(0, InpName, 0, date[d1]))
            return;
        if(!StdDevChannelPointChange(0, InpName, 1, date[d2]))
            return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
        if(IsStopped())
            return;
        //--- den Chart neu zeichnen
        ChartRedraw();
        // Verzögerung 0.05 Sekunden
        Sleep(50);
    };
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//--- Schleifenzähler
    double v_steps=InpDeviation*2;
//--- den Kanal erweitern
    for(double i=InpDeviation;i<v_steps;i+=10.0/accuracy)
    {
        if(!StdDevChannelDeviationChange(0, InpName, i))
            return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
        if(IsStopped())
            return;
        //--- den Chart neu zeichnen
        ChartRedraw();
    };
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//--- den Kanal aus dem Chart löschen
    StdDevChannelDelete(0, InpName);
    ChartRedraw();
//--- 1 Sekunde Verzögerung
    Sleep(1000);

```

```
//---  
; }
```

## OBJ\_REGRESSION

Der Kanal auf der linearen Regression.



### Hinweis

Für das Objekt "Kanal auf der linearen Regression" können Sie die weitere Anzeige nach rechts und/oder nach links (Eigenschaften [OBJPROP\\_RAY\\_RIGHT](#) und [OBJPROP\\_RAY\\_LEFT](#) jeweils) angeben. Sie können auch die Farbfüllung des Kanals angeben.

### Beispiel

Das folgende Skript erstellt und bewegt auf dem Chart den Kanal auf der linearen Regression. Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Das Skript erstellt ein graphisches Objekt \"Kanal auf der line
#property description "Koordinaten der Ankerpunkten werden in Pixeln angegeben"
#property description "Chartfenster."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="Regression"; // Kanalname
input int         InpDate1=10;          // Datum des ersten Punktes in %
input int         InpDate2=40;          // Datum des zweiten Punktes in %
input color       InpColor=clrRed;      // Kanalfarbe
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // Stil der Kanallinien
input int         InpWidth=2;           // Breite der Kanallinien
```

```

input bool      InpFill=false;      // Farbfüllung des Kanals
input bool      InpBack=false;      // Kanal im Hintergrund
input bool      InpSelection=true;  // Wählen um zu bewegen
input bool      InpRayLeft=false;   // Fortsetzung des Kanals nach links
input bool      InpRayRight=false;  // Fortsetzung des Kanals nach rechts
input bool      InpHidden=true;     // Ausgeblendet in der Objektliste
input long      InpZOrder=0;        // Priorität auf Mausklick
//+-----+
//| Erstellt Kanal auf der linearen Regression mit Koordinaten |
//+-----+
bool RegressionCreate(const long      chart_ID=0,      // ID des Charts
                     const string    name="Regression", // Kanalname
                     const int       sub_window=0,    // Nummer des Unterfens
                     datetime         time1=0,        // Zeit des ersten Punk
                     datetime         time2=0,        // Zeit des zweiten Punk
                     const color      clr=clrRed,     // Kanalfarbe
                     const ENUM_LINE_STYLE style=STYLE_SOLID, // Stil der Kanallinier
                     const int       width=1,        // Breite der Kanallini
                     const bool      fill=false,     // Farbfüllung des Kan
                     const bool      back=false,     // Im Hintergrund
                     const bool      selection=true,  // Wählen um zu beweger
                     const bool      ray_left=false, // Fortsetzung des Kan
                     const bool      ray_right=false, // Fortsetzung des Kan
                     const bool      hidden=true,    // Ausgeblendet in der
                     const long      z_order=0)      // Priorität auf Mauskli

{
//--- Die Koordinaten der Bezugspunkte angeben, wenn sie nicht gesetzt sind
    ChangeRegressionEmptyPoints(time1,time2);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Kanal auf angegebenen Koordinaten erstellen
    if(!ObjectCreate(chart_ID,name,OBJ_REGRESSION,sub_window,time1,0,time2,0))
    {
        Print(__FUNCTION__,
              ": Kanal auf der linearen Regression konnte nicht erstellt werden! Fehlercode: ",
              GetLastError());
        return(false);
    }
//--- Kanalfarbe setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- Stil der Kanallinien einstellen
    ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- Breite der Kanallinien einstellen
    ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- aktivieren (true) oder deaktivieren (false) Kanalfüllung
    ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- aktivieren (true) oder deaktivieren (false) Wählen des Kanals für Bewegung
//--- Wenn Sie ein grafisches Objekt durch Funktion ObjectCreate erstellen, das Objekt

```

```

//--- kann nicht ausgewählt und verschoben werden. Innerhalb dieser Methode ist der St
//--- true, was Sie das Objekt auswählen und verschieben erlaubt
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- aktivieren (true) oder deaktivieren (false) Fortsetzung des Kanals nach links
    ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- aktivieren (true) oder deaktivieren (false) Fortsetzung des Kanals nach rechts
    ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der
    ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- setzen die Priorität für eine Mausklick-Ereignisse auf dem Chart
    ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
    return(true);
; }

//+-----+
//| Verschiebt die Kanalbindung |
//+-----+
bool RegressionPointChange(const long   chart_ID=0,    // ID des Charts
                           const string name="Channel", // Kanalname
                           const int   point_index=0, // Nummer des Ankerpunktes
                           datetime    time=0)        // Zeitkoordinate des Ankerpunktes
{
//--- wenn die Zeit des Punktes nicht angegeben ist, bewegen wir ihn auf den letzten P
    if(!time)
        time=TimeCurrent();
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- bewegen den Bezugspunkt
    if(!ObjectMove(chart_ID,name,point_index,time,0))
    {
        Print(__FUNCTION__,
              ": der Bezugspunkt konnte nicht bewegt werden! Fehlercode = ",GetLastError());
        return(false);
    }
; }
//--- die erfolgreiche Umsetzung
    return(true);
; }

//+-----+
//| Löscht den Kanal |
//+-----+
bool RegressionDelete(const long   chart_ID=0,    // ID des Charts
                      const string name="Channel") // Kanalname
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- den Kanal löschen
    if(!ObjectDelete(chart_ID,name))
    {

```



```

        Print(__FUNCTION__,
              ": kann nicht den Kanal löschen! Fehlercode = ",GetLastError());
        return(false);
    };
}
//--- die erfolgreiche Umsetzung
return(true);
};
//+-----+
//| Überprüft die Werte der Kanalbindungspunkten für leere Werte |
//| setzt Standardwerte |
//+-----+
void ChangeRegressionEmptyPoints(datetime &time1,datetime &time2)
{
    //--- wenn die Zeit des zweiten Punktes nicht angegeben ist, wird sie auf dem aktuellen
    if(!time2)
        time2=TimeCurrent();
    //--- wenn die Zeit des ersten Punktes nicht angegeben ist, wird sie auf 9 Baren nach
    if(!time1)
    {
        //--- Array um Eröffnungszeit der letzten 10 Bars aufzunehmen
        datetime temp[10];
        CopyTime(Symbol(),Period(),time2,10,temp);
        //--- stellen wir den ersten Punkt 9 Baren links dem zweiten Punkt
        time1=temp[0];
    };
};
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    //--- Überprüfen die Richtigkeit der Eingabeparameter
    if(InpDate1<0 || InpDate1>100 ||
       InpDate2<0 || InpDate2>100)
    {
        Print("Fehler! Ungültige Eingabeparameter! ");
        return;
    };
    //--- Die Anzahl der sichtbaren Bars im Chart-Fenster
    int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
    //--- Größe des Arrays price
    int accuracy=1000;
    //--- Arrays für Werte von Daten und Preise, die für Setzung und Änderung
    //--- der Koordinaten der Kanalbindungspunkten verwendet werden
    datetime date[];
    double price[];
    //--- Speicher reservieren
    ArrayResize(date,bars);
    ArrayResize(price,accuracy);

```

```

//--- Füllen die Anordnung von Daten
ResetLastError();
if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
{
    Print("Kann nicht den Wert der Zeit kopieren! Fehlercode = ",GetLastError());
    return;
};
//--- Füllen den Array der Preise
//--- den Minimal- und Maximalwert der Charts finden
double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- Schritte der Preisänderung finden und das Array füllen
double step=(max_price-min_price)/accuracy;
for(int i=0;i<accuracy;i++)
    price[i]=min_price+i*step;
//--- Definieren die Punkte um den Kanal zu zeichnen
int d1=InpDate1*(bars-1)/100;
int d2=InpDate2*(bars-1)/100;
//--- Einen Kanal auf der linearen Regression erstellen
if(!RegressionCreate(0,InpName,0,date[d1],date[d2],InpColor,InpStyle,InpWidth,
    InpFill,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder))
{
    return;
};
//--- den Chart neu zeichnen und 1 Sekunde warten
ChartRedraw();
Sleep(1000);
//--- jetzt bewegen wir den Kanal auf der linearen Regression horizontal nach rechts
//--- Schleifenzähler
int h_steps=bars/2;
//--- Kanal bewegen
for(int i=0;i<h_steps;i++)
{
    //--- die nächsten Werte nehmen
    if(d1<bars-1)
        d1+=1;
    if(d2<bars-1)
        d2+=1;
    //--- Ankerpunkte bewegen
    if(!RegressionPointChange(0,InpName,0,date[d1]))
        return;
    if(!RegressionPointChange(0,InpName,1,date[d2]))
        return;
}
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
if(IsStopped())
    return;
//--- den Chart neu zeichnen
ChartRedraw();
// Verzögerung 0.05 Sekunden

```

```
        Sleep(50);  
    };  
    //--- 1 Sekunde Verzögerung  
    Sleep(1000);  
    //--- den Kanal aus dem Chart löschen  
    RegressionDelete(0, InpName);  
    ChartRedraw();  
    //--- 1 Sekunde Verzögerung  
    Sleep(1000);  
    //---  
};
```

## OBJ\_PITCHFORK

Andrew's Pitchfork.



### Hinweis

Für das Objekt "Andrew's Pitchfork" können Sie die weitere Anzeige nach rechts und/oder nach links (Eigenschaften [OBJPROP\\_RAY\\_RIGHT](#) und [OBJPROP\\_RAY\\_LEFT](#) jeweils) angeben.

Sie können auch die Anzahl der Linien-Ebenen, ihre Werte und Farben angeben.

### Beispiel

Das folgende Skript erstellt und bewegt auf dem Chart Objekt "Andrew's Pitchfork". Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Das Skript erstellt ein graphisches Objekt \"Andrew's Pitchfork\"
#property description "Koordinaten des Bezugspunktes werden in Prozent von"
#property description "der Größe des Chartfensters eingegeben."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="Pitchfork";    // Name
input int         InpDate1=14;            // Datum des ersten Punktes in %
input int         InpPrice1=40;           // Preis des ersten Punktes in %
input int         InpDate2=18;            // Datum des zweiten Punktes in %
input int         InpPrice2=50;           // Preis des zweiten Punktes in %
input int         InpDate3=18;            // Datum des dritten Punktes in %
input int         InpPrice3=30;           // Preis des dritten Punktes in %
```

```

input color      InpColor=clrRed;      // Farbe
input ENUM_LINE_STYLE InpStyle=STYLE_SOLID; // Linienstil
input int        InpWidth=1;          // Linienbreite
input bool       InpBack=false;       // Pitchfork im Hintergrund
input bool       InpSelection=true;    // Wählen um zu bewegen
input bool       InpRayLeft=false;    // Fortsetzung von Pitchfork nach links
input bool       InpRayRight=false;   // Fortsetzung von Pitchfork nach rechts
input bool       InpHidden=true;      // Ausgeblendet in der Objektliste
input long       InpZOrder=0;         // Priorität auf Mausclick
//+-----+
//| Erstellt Andrew's Pitchfork auf angegebenen Koordinaten |
//+-----+
bool PitchforkCreate(const long      chart_ID=0,      // ID des Charts
                    const string    name="Pitchfork", // Name
                    const int       sub_window=0,    // Nummer des Unterfensters
                    datetime         time1=0,        // Zeit des ersten Punktes
                    double            price1=0,       // Preis des ersten Punktes
                    datetime         time2=0,        // Zeit des zweiten Punktes
                    double            price2=0,       // Preis des zweiten Punktes
                    datetime         time3=0,        // Zeit des dritten Punktes
                    double            price3=0,       // Preis des dritten Punktes
                    const color      clr=clrRed,     // Linienfarbe
                    const ENUM_LINE_STYLE style=STYLE_SOLID, // Linienstil
                    const int       width=1,        // Linienbreite
                    const bool       back=false,     // In der Hintergrund
                    const bool       selection=true, // Wählen um zu bewegen
                    const bool       ray_left=false, // Fortsetzung von Pitchfork nach links
                    const bool       ray_right=false, // Fortsetzung von Pitchfork nach rechts
                    const bool       hidden=true,    // Verborgen in der Liste der Objekte
                    const long      z_order=0)      // Priorität für Mausclick
{
//--- Die Koordinaten der Bezugspunkte angeben, wenn sie nicht gesetzt sind
    ChangeChannelEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Andrew's Pitchfork auf angegebenen Koordinaten erstellen
    if(!ObjectCreate(chart_ID,name,OBJ_PITCHFORK,sub_window,time1,price1,time2,price2,time3,price3))
    {
        Print(__FUNCTION__,
              ": Objekt \" Andrew's Pitchfork\" konnte nicht erstellt werden! Fehlercode: ",
              GetLastError(),
              "\n");
        return(false);
    }
//--- Farbe setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- Linienstil setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- Linienbreite angeben
    ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- Im Vordergrund (false) oder Hintergrund (true) anzeigen

```

```

    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- aktivieren (true) oder deaktivieren (false) Wählen von Andrew's Pitchfork für Be
//--- Wenn Sie ein grafisches Objekt durch Funktion ObjectCreate erstellen, das Objekt
//--- kann nicht ausgewählt und verschoben werden. Innerhalb dieser Methode ist der St
//--- true, was Sie das Objekt auswählen und verschieben erlaubt
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- aktivieren (true) oder deaktivieren (false) Fortsetzung von Andrew's Pitchfork r
    ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- aktivieren (true) oder deaktivieren (false) Fortsetzung von Andrew's Pitchfork r
    ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der
    ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- setzen die Priorität für eine Mausklick-Ereignisse auf dem Chart
    ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Gibt Anzahl der Niveaus von "Andrew's Pitchfork" und Parameter an|
//+-----+
bool PitchforkLevelsSet(int          levels,          // Anzahl der Niveaulinien
                        double        &values[],       // Werte der Linien
                        color         &colors[],       // Farbe der Linien
                        ENUM_LINE_STYLE &styles[],      // Linienstil
                        int            &widths[],       // Linienbreite
                        const long     chart_ID=0,      // ID des Charts
                        const string   name="Pitchfork") // Name
{
//--- Überprüfen wir die Größe des Arrays
    if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
        levels!=ArraySize(widths) || levels!=ArraySize(widths))
    {
        Print(__FUNCTION__,": Länge des Arrays entspricht nicht der Anzahl der Niveaus,
        return(false);
    }
};
//--- Die Anzahl der Ebenen angeben
    ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- die Eigenschaften der Ebenen im Zyklus angeben
    for(int i=0;i<levels;i++)
    {
        //--- Wert der Ebene
        ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
        //--- Farbe der Ebene
        ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
        //--- Stil der Ebene
        ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
        //--- Stärke der Ebene
        ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
    }
}

```

```

        //--- Beschreibung der Ebene
        ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
    );
//--- die erfolgreiche Umsetzung
    return(true);
};

//+-----+
//| Verschiebt den Ankerpunkt von Andrew's Pitchfork |
//+-----+
bool PitchforkPointChange(const long   chart_ID=0,      // ID des Charts
                          const string name="Pitchfork", // Kanalname
                          const int    point_index=0,   // Nummer des Ankerpunktes
                          datetime     time=0,          // Zeitkoordinate des Punktes
                          double       price=0)         // Preiskoordinate des Punktes
{
//--- Wenn die Koordinaten des Punktes nicht angegeben sind, dann verschieben wir es a
    if(!time)
        time=TimeCurrent();
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- bewegen den Bezugspunkt
    if(!ObjectMove(chart_ID,name,point_index,time,price))
    {
        Print(__FUNCTION__,
              ": der Bezugspunkt konnte nicht bewegt werden! Fehlercode = ",GetLastError()
        );
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
};

//+-----+
//| Löscht das Objekt "Andrew's Pitchfork" |
//+-----+
bool PitchforkDelete(const long   chart_ID=0,      // ID des Charts
                     const string name="Pitchfork") // Kanalname
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- den Kanal löschen
    if(!ObjectDelete(chart_ID,name))
    {
        Print(__FUNCTION__,
              ": Objekt \" Andrew's Pitchfork\" konnte nicht gelöscht werden! Fehlercode
        );
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
};

```

```

; }
//+-----+
//| Überprüft die Werte der Ankerpunkten von Andrew's Pitchfork |
//| und setzt Standardwerte für leere Werte |
//+-----+
void ChangeChannelEmptyPoints(datetime &time1,double &price1,datetime &time2,
                             double &price2,datetime &time3,double &price3)
{
//--- wenn die Zeit des zweiten (rechten oberen) Punktes nicht angegeben ist, wird sie
    if(!time2)
        time2=TimeCurrent();
//--- wenn der Preis des zweiten Punktes nicht angegeben ist, wird er dem Bid-Wert gle
    if(!price2)
        price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- wenn die Zeit des ersten (linken) Punktes nicht angegeben ist, wird sie auf 9 Ba
    if(!time1)
    {
        //--- Array um Eröffnungszeit der letzten 10 Bars aufzunehmen
        datetime temp[10];
        CopyTime(Symbol(),Period(),time2,10,temp);
        //--- stellen wir den ersten Punkt 9 Baren links dem zweiten Punkt
        time1=temp[0];
    }
; }
//--- wenn der Preis des ersten Punktes nicht angegeben ist, bewegen wir ihn 200 Punkt
    if(!price1)
        price1=price2-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- wenn die Zeit des dritten Punktes nicht angegeben ist, wird sie der Zeit des zwe
    if(!time3)
        time3=time2;
//--- wenn der Preis des dritten Punktes nicht angegeben ist, bewegen wir ihn 200 Punt
    if(!price3)
        price3=price1-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
; }
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//--- Überprüfen die Richtigkeit der Eingabeparameter
    if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
        InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
        InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
    {
        Print("Fehler! Ungültige Eingabeparameter! ");
        return;
    }
; }
//--- Die Anzahl der sichtbaren Bars im Chart-Fenster
    int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- Größe des Arrays price

```



```

    int accuracy=1000;
    //--- Arrays für Werte von Daten und Preise, die für Setzung und Änderung
    //--- der Koordinaten der Ankerpunkten von Andrew's Pitchfork verwendet werden
    datetime date[];
    double price[];
    //--- Speicher reservieren
    ArrayResize(date,bars);
    ArrayResize(price,accuracy);
    //--- Füllen die Anordnung von Daten
    ResetLastError();
    if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
    {
        Print("Kann nicht den Wert der Zeit kopieren! Fehlercode = ",GetLastError());
        return;
    }
    //--- Füllen den Array der Preise
    //--- den Minimal- und Maximalwert der Charts finden
    double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
    double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
    //--- Schritte der Preisänderung finden und das Array füllen
    double step=(max_price-min_price)/accuracy;
    for(int i=0;i<accuracy;i++)
        price[i]=min_price+i*step;
    //--- Definieren die Punkte um Andrew's Pitchfork zu zeichnen
    int d1=InpDate1*(bars-1)/100;
    int d2=InpDate2*(bars-1)/100;
    int d3=InpDate3*(bars-1)/100;
    int p1=InpPrice1*(accuracy-1)/100;
    int p2=InpPrice2*(accuracy-1)/100;
    int p3=InpPrice3*(accuracy-1)/100;
    //--- Andrew's Pitchfork erstellen
    if(!PitchforkCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],price[p3],
        InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden))
    {
        return;
    }
    //--- den Chart neu zeichnen und 1 Sekunde warten
    ChartRedraw();
    Sleep(1000);
    //--- jetzt bewegen wir die Ankerpunkte
    //--- Schleifenzähler
    int v_steps=accuracy/10;
    //--- den ersten Punkt bewegen
    for(int i=0;i<v_steps;i++)
    {
        //--- den nächsten Wert nehmen
        if(p1>1)
            p1-=1;
        //--- den Punkt bewegen
    }

```

```

        if(!PitchforkPointChange(0, InpName, 0, date[d1], price[p1]))
            return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
        if(IsStopped())
            return;
//--- den Chart neu zeichnen
        ChartRedraw();
    };
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//--- Schleifenzähler
    int h_steps=bars/8;
//--- den dritten Punkt bewegen
    for(int i=0; i<h_steps; i++)
    {
        //--- den nächsten Wert nehmen
        if(d3<bars-1)
            d3+=1;
        //--- den Punkt bewegen
        if(!PitchforkPointChange(0, InpName, 2, date[d3], price[p3]))
            return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
        if(IsStopped())
            return;
//--- den Chart neu zeichnen
        ChartRedraw();
//--- den Chart neu zeichnen
        ChartRedraw();
// Verzögerung 0.05 Sekunden
        Sleep(50);
    };
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//--- Schleifenzähler
    v_steps=accuracy/10;
//--- den zweiten Punkt bewegen
    for(int i=0; i<v_steps; i++)
    {
        //--- den nächsten Wert nehmen
        if(p2>1)
            p2-=1;
        //--- den Punkt bewegen
        if(!PitchforkPointChange(0, InpName, 1, date[d2], price[p2]))
            return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
        if(IsStopped())
            return;
//--- den Chart neu zeichnen
        ChartRedraw();

```

```
    ; }  
    //--- 1 Sekunde Verzögerung  
    Sleep(1000);  
    //--- Andrew's Pitchfork aus dem Chart löschen  
    PitchforkDelete(0, InpName);  
    ChartRedraw();  
    //--- 1 Sekunde Verzögerung  
    Sleep(1000);  
    //---  
    ; }
```

## OBJ\_GANNLINIE

Gann Linie.



### Hinweis

Für das Objekt "Gann Linie" können Sie die weitere Anzeige nach rechts und/oder nach links (Eigenschaften [OBJPROP\\_RAY\\_RIGHT](#) und [OBJPROP\\_RAY\\_LEFT](#) jeweils) angeben.

Um die Linienneigung anzugeben, können Sie Gann Winkel mit dem Maßstab und Koordinaten des zweiten Ankerpunktes verwenden.

### Beispiel

Das folgende Skript erstellt und bewegt auf dem Chart Objekt "Gann Linie". Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Das Skript erstellt ein graphisches Objekt \"Gann Linie\"."
#property description "Koordinaten des Bezugspunktes werden in Prozent von"
#property description "der Größe des Chartfensters eingegeben."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="GannLine";           // Liniennamen
input int         InpDate1=20;                  // Datum des ersten Punktes in %
input int         InpPrice1=75;                 // Preis des ersten Punktes in %
input int         InpDate2=80;                 // Datum des zweiten Punktes in %
input double      InpAngle=0.0;                 // Gann Winkel
```

```

input double      InpScale=1.0;           // Maßstab
input color       InpColor=clrRed;        // Linienfarbe
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Linienstil
input int         InpWidth=2;             // Linienbreite
input bool        InpBack=false;          // Die Linie im Hintergrund
input bool        InpSelection=true;       // Wählen um zu bewegen
input bool        InpRayLeft=false;        // Fortsetzung der Linie nach links
input bool        InpRayRight=true;        // Fortsetzung der Linie nach rechts
input bool        InpHidden=true;         // Ausgeblendet in der Objektliste
input long        InpZOrder=0;            // Priorität auf Mausклик

//+-----+
//| Erstellt Gann Linie auf Koordinaten, Winkel und Maßstab |
//+-----+

bool GannLineCreate(const long      chart_ID=0,      // ID des Charts
                    const string    name="GannLine", // Liniennamen
                    const int       sub_window=0,    // Nummer des Fensters
                    datetime        time1=0,         // Zeit des ersten Punktes
                    double          price1=0,        // Preis des ersten Punktes
                    datetime        time2=0,         // Zeit des zweiten Punktes
                    const double     angle=1.0,       // Gann Winkel
                    const double     scale=1.0,       // Maßstab
                    const color      clr=clrRed,      // Linienfarbe
                    const ENUM_LINE_STYLE style=STYLE_SOLID, // Linienstil
                    const int        width=1,         // Linienbreite
                    const bool        back=false,     // In der Hintergrund
                    const bool        selection=true,  // Wählen um zu bewegen
                    const bool        ray_left=false, // Fortsetzung der Linie
                    const bool        ray_right=true, // Fortsetzung der Linie
                    const bool        hidden=true,    // Verborgen in der Liste
                    const long        z_order=0)      // Priorität für Mausклик
{
    //--- Die Koordinaten der Bezugspunkte angeben, wenn sie nicht gesetzt sind
    ChangeGannLineEmptyPoints(time1,price1,time2);
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- "Gann Linie" auf angegebenen Koordinaten erstellen
    //--- Richtige Preiskoordinate des zweiten Ankerpunktes wird automatisch
    //--- neu gesetzt nachdem Gann Winkel und (oder) Maßstab verändert ist,
    if(!ObjectCreate(chart_ID,name,OBJ_GANNLINIE,sub_window,time1,price1,time2,0))
    {
        Print(__FUNCTION__,
              ": Objekt \" Gann Linie\" konnte nicht erstellt werden! Fehlercode = ",GetLastError());
        return(false);
    }
    ;}
    //--- Gann Winkel ändern
    ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);
    //--- Ändern wir den Maßstab (Anzahl der Pips pro Bar)
    ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
    //--- Farbe der Linie setzen

```

```

    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- Linienstil setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- Linienbreite setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- aktivieren (true) oder deaktivieren (false) Wählen der Linie für Bewegung
//--- Wenn Sie ein grafisches Objekt durch Funktion ObjectCreate erstellen, das Objekt
//--- kann nicht ausgewählt und verschoben werden. Innerhalb dieser Methode ist der St
//--- true, was Sie das Objekt auswählen und verschieben erlaubt
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- aktivieren (true) oder deaktivieren (false) Fortsetzung der Linie nach links
    ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- aktivieren (true) oder deaktivieren (false) Fortsetzung der Linie nach rechts
    ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der
    ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- setzen die Priorität für eine Mausklick-Ereignisse auf dem Chart
    ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
    return(true);
; }
//+-----+
//| Verschiebt den Ankerpunkt von Gann Linien |
//+-----+
bool GannLinePointChange(const long   chart_ID=0,      // ID des Charts
                        const string name="GannLine",  // Liniennamen
                        const int    point_index=0,    // Nummer des Ankerpunktes
                        datetime     time=0,          // Zeitkoordinate des Punktes
                        double        price=0)         // Preiskoordinate des Punktes
{
//--- Wenn die Koordinaten des Punktes nicht angegeben sind, dann verschieben wir es
    if(!time)
        time=TimeCurrent();
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- bewegen den Bezugspunkt der Linie
    if(!ObjectMove(chart_ID,name,point_index,time,price))
    {
        Print(__FUNCTION__,
              ": der Bezugspunkt konnte nicht bewegt werden! Fehlercode = ",GetLastError());
        return(false);
    }
; }
//--- die erfolgreiche Umsetzung
    return(true);

```

```

; }
//+-----+
//| Ändert Gann Winkel |
//+-----+
bool GannLineAngleChange(const long   chart_ID=0,      // ID des Charts
                        const string name="GannLine",  // Liniennamen
                        const double angle=1.0)        // Gann Winkel
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Gann Winkel ändern
    if(!ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle))
    {
        Print(__FUNCTION__,
              ": Kann den Gann Winkel nicht ändern! Fehlercode = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
}
//+-----+
//| Ändert den Maßstab von "Gann Linie" |
//+-----+
bool GannLineScaleChange(const long   chart_ID=0,      // ID des Charts
                        const string name="GannLine",  // Liniennamen
                        const double scale=1.0)        // Maßstab
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Ändern wir den Maßstab (Anzahl der Pips pro Bar)
    if(!ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale))
    {
        Print(__FUNCTION__,
              ": Konnte nicht den Maßstab ändern! Fehlercode = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
}
//+-----+
//| Die Funktion löscht das Objekt "Gann Linie" |
//+-----+
bool GannLineDelete(const long   chart_ID=0,      // ID des Charts
                   const string name="GannLine") // Liniennamen
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Gann Linie löschen
    if(!ObjectDelete(chart_ID,name))

```

```

    {
        Print(__FUNCTION__,
            ": Objekt \" Gann Linie\" konnte nicht gelöscht werden! Fehlercode = ",GetLastError(),
            return(false);
    };
}

//--- die erfolgreiche Umsetzung
return(true);
};

//+-----+
//| Überprüft die Werte der Ankerpunkten von Gann Linie |
//| und setzt Standardwerte für leere Werte |
//+-----+
void ChangeGannLineEmptyPoints(datetime &time1,double &price1,datetime &time2)
{
    //--- wenn die Zeit des zweiten Punktes nicht angegeben ist, wird sie auf dem aktuellen
    if(!time2)
        time2=TimeCurrent();
    //--- wenn die Zeit des ersten Punktes nicht angegeben ist, wird sie auf 9 Baren nach
    if(!time1)
    {
        //--- Array um Eröffnungszeit der letzten 10 Bars aufzunehmen
        datetime temp[10];
        CopyTime(Symbol(),Period(),time2,10,temp);
        //--- stellen wir den ersten Punkt 9 Baren links dem zweiten Punkt
        time1=temp[0];
    };
    //--- Wenn der Preis des ersten Punktes nicht angegeben wird, dann wird es einen Wert
    if(!price1)
        price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
    };
    //+-----+
    //| Script program start function |
    //+-----+
    void OnStart()
    {
        //--- Überprüfen die Richtigkeit der Eingabeparameter
        if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
            InpDate2<0 || InpDate2>100)
        {
            Print("Fehler! Ungültige Eingabeparameter! ");
            return;
        };
    };
    //--- Die Anzahl der sichtbaren Bars im Chart-Fenster
    int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
    //--- Größe des Arrays price
    int accuracy=1000;
    //--- Arrays für Werte von Daten und Preise, die für Setzung und Änderung
    //--- der Koordinaten von Bezugspunkte der Linie verwendet werden
    datetime date[];

```



```

    double price[];
//--- Speicher reservieren
    ArrayResize(date,bars);
    ArrayResize(price,accuracy);
//--- Füllen die Anordnung von Daten
    ResetLastError();
    if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
    {
        Print("Kann nicht den Wert der Zeit kopieren! Fehlercode = ",GetLastError());
        return;
    }
//--- Füllen den Array der Preise
//--- den Minimal- und Maximalwert der Charts finden
    double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
    double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- Schritte der Preisänderung finden und das Array füllen
    double step=(max_price-min_price)/accuracy;
    for(int i=0;i<accuracy;i++)
        price[i]=min_price+i*step;
//--- Definieren die Punkte um Gann Linie zu zeichnen
    int d1=InpDate1*(bars-1)/100;
    int d2=InpDate2*(bars-1)/100;
    int p1=InpPrice1*(accuracy-1)/100;
//--- Gann Linie erstellen
    if(!GannLineCreate(0,InpName,0,date[d1],price[p1],date[d2],InpAngle,InpScale,InpColor,
        InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder))
    {
        return;
    }
//--- den Chart neu zeichnen und 1 Sekunde warten
    ChartRedraw();
    Sleep(1000);
//--- jetzt bewegen wir den Ankerpunkt von der Linie und ändern den Winkel
//--- Schleifenzähler
    int v_steps=accuracy/2;
//--- Wir bewegen den ersten Bezugspunkt vertikal
    for(int i=0;i<v_steps;i++)
    {
        //--- den nächsten Wert nehmen
        if(p1>1)
            p1-=1;
        //--- den Punkt bewegen
        if(!GannLinePointChange(0,InpName,0,date[d1],price[p1]))
            return;
    }
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
    if(IsStopped())
        return;
//--- den Chart neu zeichnen
    ChartRedraw();

```

```
    ;}

//--- Halb-Sekunden-Verzögerung
    Sleep(500);

//--- Definieren wir den aktuellen wert des Gann-Winkels (hat
//--- nach der Bewegung des ersten Punktes geändert)
    double curr_angle;
    if(!ObjectGetDouble(0, InpName, OBJPROP_ANGLE, 0, curr_angle))
        return;

//--- Schleifenzähler
    v_steps=accuracy/8;
//--- Gann Winkel ändern
    for(int i=0; i<v_steps; i++)
    {
        if(!GannLineAngleChange(0, InpName, curr_angle-0.05*i))
            return;

//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
        if(IsStopped())
            return;

//--- den Chart neu zeichnen
        ChartRedraw();
    }

//--- 1 Sekunde Verzögerung
    Sleep(1000);

//--- die Linie aus dem Chart löschen
    GannLineDelete(0, InpName);
    ChartRedraw();

//--- 1 Sekunde Verzögerung
    Sleep(1000);

//---
    ;}
```

## OBJ\_GANNFAN

Gann Fan.



### Hinweis

Für "Gann Fan" können Sie den Trendtyp aus der Aufzählung [ENUM\\_GANN\\_DIRECTION](#) angeben. Durch die Anpassung des Maßstabs ([OBJPROP\\_SCALE](#)), können Sie den Winkel der Fanlinien ändern.

### Beispiel

Das folgende Skript erstellt und bewegt auf dem Chart Objekt "Gann Fan". Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Das Skript erstellt ein graphisches Objekt \"Gann Fan\"."
#property description "Koordinaten des Bezugspunktes werden in Prozent von"
#property description "der Größe des Chartfensters eingegeben."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="GannFan";           // Name
input int         InpDate1=15;                 // Datum des ersten Punktes in %
input int         InpPrice1=25;                // Preis des ersten Punktes in %
input int         InpDate2=85;                // Datum des zweiten Punktes in %
input double      InpScale=2.0;                // Maßstab
input bool        InpDirection=false;         // Trendrichtung
input color       InpColor=clrRed;             // Farbe von Fan
```

```

input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Stil der Fanlinien
input int              InpWidth=1;               // Breite der Fanlinien
input bool             InpBack=false;            // Fan im Hintergrund
input bool             InpSelection=true;        // Wählen um zu bewegen
input bool             InpHidden=true;          // Ausgeblendet in der Objektliste
input long             InpZOrder=0;             // Priorität auf Mausclick

//+-----+
//| Erstellt "Gann Fan"                                     |
//+-----+

bool GannFanCreate(const long      chart_ID=0,      // ID des Charts
                  const string    name="GannFan",  // Name des Fans
                  const int       sub_window=0,    // Nummer des Unterfensters
                  datetime        time1=0,         // Zeit des ersten Punktes
                  double          price1=0,        // Preis des ersten Punktes
                  datetime        time2=0,         // Zeit des zweiten Punktes
                  const double    scale=1.0,      // Maßstab
                  const bool      direction=true,  // Trendrichtung
                  const color     clr=clrRed,     // Fanfarbe
                  const ENUM_LINE_STYLE style=STYLE_SOLID, // Stil der Fanlinien
                  const int       width=1,        // Breite der Fanlinien
                  const bool      back=false,     // Im Hintergrund
                  const bool      selection=true,  // Auswählen um zu bewegen
                  const bool      hidden=true,    // Ausgeblendet in der Objektliste
                  const long      z_order=0)      // Priorität auf Mausclick
{
    //--- Die Koordinaten der Bezugspunkte angeben, wenn sie nicht gesetzt sind
    ChangeGannFanEmptyPoints(time1,price1,time2);
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- "Gann Fan" auf angegebenen Koordinaten erstellen
    if(!ObjectCreate(chart_ID,name,OBJ_GANNFAN,sub_window,time1,price1,time2,0))
    {
        Print(__FUNCTION__,
              ": Objekt \" Gann Fan\" konnte nicht erstellt werden! Fehlercode = ",GetLastError());
        return(false);
    }
    ;}

    //--- Ändern wir den Maßstab (Anzahl der Pips pro Bar)
    ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
    //--- Ändern wir die Richtung des Trends von "Gann Fan" (true - absteigend, falsch - aufsteigend)
    ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction);
    //--- Fanfarbe setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
    //--- Linienstil angeben
    ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
    //--- Breite der Fanlinien einstellen
    ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
    //--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
    //--- aktivieren (true) oder deaktivieren (false) Wählen von Fan für Bewegung

```

```

//--- Wenn Sie ein grafisches Objekt durch Funktion ObjectCreate erstellen, das Objekt
//--- kann nicht ausgewählt und verschoben werden. Innerhalb dieser Methode ist der St
//--- true, was Sie das Objekt auswählen und verschieben erlaubt
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der
    ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- setzen die Priorität für eine Mausklick-Ereignisse auf dem Chart
    ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
    return(true);
; }

//+-----+
//| Verschiebt den Ankerpunkt von Gann Fan |
//+-----+
bool GannFanPointChange(const long   chart_ID=0,    // ID des Charts
                        const string name="GannFan", // Name
                        const int    point_index=0, // Nummer des Punktes
                        datetime      time=0,       // Zeitkoordinate des Punktes
                        double         price=0)      // Preiskoordinate des Punktes
{
//--- Wenn die Koordinaten des Punktes nicht angegeben sind, dann verschieben wir es
    if(!time)
        time=TimeCurrent();
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Ankerpunkt von Fan bewegen
    if(!ObjectMove(chart_ID,name,point_index,time,price))
    {
        Print(__FUNCTION__,
              ": der Bezugspunkt konnte nicht bewegt werden! Fehlercode = ",GetLastError());
        return(false);
    }
; }
//--- die erfolgreiche Umsetzung
    return(true);
; }

//+-----+
//| Ändert den Maßstab von "Gann Fan" |
//+-----+
bool GannFanScaleChange(const long   chart_ID=0,    // ID des Charts
                        const string name="GannFan", // Name
                        const double scale=1.0)      // Maßstab
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Ändern wir den Maßstab (Anzahl der Pips pro Bar)
    if(!ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale))

```

```

    {
        Print(__FUNCTION__,
            ": Konnte nicht den Maßstab ändern! Fehlercode = ",GetLastError());
        return(false);
    }
}
//--- die erfolgreiche Umsetzung
return(true);
; }

//+-----+
//| Ändert die Trendrichtung von "Gann Fan" |
//+-----+
bool GannFanDirectionChange(const long   chart_ID=0,    // ID des Charts
                           const string name="GannFan", // Name
                           const bool   direction=true) // Trendrichtung
{
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Ändern wir die Trendrichtung von "Gann Fan"
    if(!ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction))
    {
        Print(__FUNCTION__,
            ": Konnte nicht die Trendrichtung ändern! Fehlercode = ",GetLastError());
        return(false);
    }
}
//--- die erfolgreiche Umsetzung
return(true);
; }

//+-----+
//| Die Funktion löscht das Objekt "Gann Fan" |
//+-----+
bool GannFanDelete(const long   chart_ID=0,    // ID des Charts
                  const string name="GannFan") // Name
{
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Gann Fan löschen
    if(!ObjectDelete(chart_ID,name))
    {
        Print(__FUNCTION__,
            ": Objekt \" Gann Fan\" konnte nicht gelöscht werden! Fehlercode = ",GetLastError());
        return(false);
    }
}
//--- die erfolgreiche Umsetzung
return(true);
; }

//+-----+
//| Überprüft die Werte der Ankerpunkten von Gann Fan |
//| und setzt Standardwerte für leere Werte |
//+-----+

```

```

void ChangeGannFanEmptyPoints(datetime &time1,double &price1,datetime &time2)
{
//--- wenn die Zeit des zweiten Punktes nicht angegeben ist, wird sie auf dem aktuellen
    if(!time2)
        time2=TimeCurrent();
//--- wenn die Zeit des ersten Punktes nicht angegeben ist, wird sie auf 9 Baren nach
    if(!time1)
    {
        //--- Array um Eröffnungszeit der letzten 10 Bars aufzunehmen
        datetime temp[10];
        CopyTime(Symbol(),Period(),time2,10,temp);
        //--- stellen wir den ersten Punkt 9 Baren links dem zweiten Punkt
        time1=temp[0];
    }
//--- Wenn der Preis des ersten Punktes nicht angegeben wird, dann wird es einen Wert
    if(!price1)
        price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
};

//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//--- Überprüfen die Richtigkeit der Eingabeparameter
    if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
        InpDate2<0 || InpDate2>100)
    {
        Print("Fehler! Ungültige Eingabeparameter! ");
        return;
    }
//--- Die Anzahl der sichtbaren Bars im Chart-Fenster
    int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- Größe des Arrays price
    int accuracy=1000;
//--- Arrays für Werte von Daten und Preise, die für Setzung und Änderung
//--- der Koordinaten der Ankerpunkten von Fan verwendet werden
    datetime date[];
    double price[];
//--- Speicher reservieren
    ArrayResize(date,bars);
    ArrayResize(price,accuracy);
//--- Füllen die Anordnung von Daten
    ResetLastError();
    if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
    {
        Print("Kann nicht den Wert der Zeit kopieren! Fehlercode = ",GetLastError());
        return;
    }
//--- Füllen den Array der Preise

```

```

//--- den Minimal- und Maximalwert der Charts finden
double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- Schritte der Preisänderung finden und das Array füllen
double step=(max_price-min_price)/accuracy;
for(int i=0;i<accuracy;i++)
    price[i]=min_price+i*step;
//--- Definieren die Punkte um Gann Fan zu zeichnen
int d1=InpDate1*(bars-1)/100;
int d2=InpDate2*(bars-1)/100;
int p1=InpPrice1*(accuracy-1)/100;
//--- Gann Fan erstellen
if(!GannFanCreate(0,InpName,0,date[d1],price[p1],date[d2],InpScale,InpDirection,
    InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
{
    return;
};
//--- den Chart neu zeichnen und 1 Sekunde warten
ChartRedraw();
Sleep(1000);
//--- jetzt bewegen wir den Ankerpunkt von Fan
//--- Schleifenzähler
int v_steps=accuracy/2;
//--- Wir bewegen den ersten Bezugspunkt vertikal
for(int i=0;i<v_steps;i++)
{
    //--- den nächsten Wert nehmen
    if(p1<accuracy-1)
        p1+=1;
    //--- den Punkt bewegen
    if(!GannFanPointChange(0,InpName,0,date[d1],price[p1]))
        return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
    if(IsStopped())
        return;
    //--- den Chart neu zeichnen
    ChartRedraw();
};
//--- 1 Sekunde Verzögerung
Sleep(1000);
//--- Ändern wir die Trendrichtung von "Gann Fan" zu absteigend
GannFanDirectionChange(0,InpName,true);
//--- den Chart neu zeichnen
ChartRedraw();
//--- 1 Sekunde Verzögerung
Sleep(1000);
//--- den Fan aus dem Chart löschen
GannFanDelete(0,InpName);
ChartRedraw();

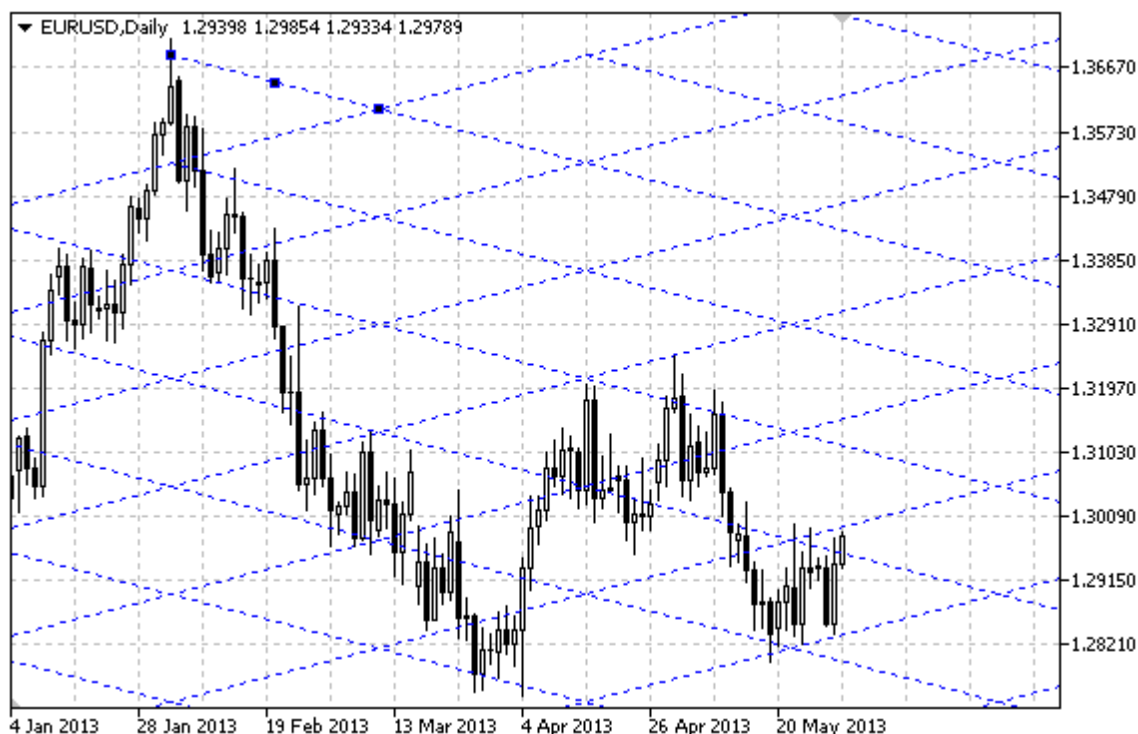
```



```
//--- 1 Sekunde Verzögerung  
    Sleep(1000);  
//---  
; }
```

## OBJ\_GANNGRID

Gann Gitter.



### Hinweis

Für "Gann Gitter" können Sie den Trendtyp aus der Aufzählung [ENUM\\_GANN\\_DIRECTION](#) angeben. Durch die Anpassung des Maßstabs ([OBJPROP\\_SCALE](#)), können Sie den Winkel der Gitterlinien ändern.

### Beispiel

Das folgende Skript erstellt und bewegt auf dem Chart Objekt "Gann Gitter". Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Das Skript erstellt ein graphisches Objekt \"Gann Gitter\"."
#property description "Koordinaten der Gitterankerpunkten werden in Prozente"
#property description "der Größe des Chartfensters eingegeben."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="GannGrid";           // Gittername
input int         InpDate1=15;                  // Datum des ersten Punktes in %
input int         InpPrice1=25;                 // Preis des ersten Punktes in %
input int         InpDate2=35;                  // Datum des zweiten Punktes in %
input double      InpScale=3.0;                 // Maßstab
input bool        InpDirection=false;          // Trendrichtung
```

```

input color      InpColor=clrRed;           // Gitterfarbe
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Gitterlinienstil
input int        InpWidth=1;               // Breite der Fanlinien
input bool       InpBack=false;            // Das Gitter im Hintergrund
input bool       InpSelection=true;        // Wählen um zu bewegen
input bool       InpHidden=true;          // Ausgeblendet in der Objektliste
input long       InpZOrder=0;              // Priorität auf Mausclick
//+-----+
//| Erstellt "Gann Gitter"                  |
//+-----+
bool GannGridCreate(const long      chart_ID=0,      // ID des Charts
                    const string   name="GannGrid",  // Gittername
                    const int      sub_window=0,    // Nummer des Fensters
                    datetime       time1=0,         // Zeit des ersten Punktes
                    double         price1=0,        // Preis des ersten Punktes
                    datetime       time2=0,         // Zeit des zweiten Punktes
                    const double   scale=1.0,       // Maßstab
                    const bool     direction=true,  // Trendrichtung
                    const color    clr=clrRed,     // Gitterfarbe
                    const ENUM_LINE_STYLE style=STYLE_SOLID, // Gitterlinienstil
                    const int      width=1,        // Gitterlinienbreite
                    const bool     back=false,     // In der Hintergrund
                    const bool     selection=true,  // Wählen um zu bewegen
                    const bool     hidden=true,    // Verborgen in der Liste
                    const long     z_order=0)      // Priorität für Mausclick
{
    //--- Die Koordinaten der Bezugspunkte angeben, wenn sie nicht gesetzt sind
    ChangeGannGridEmptyPoints(time1,price1,time2);
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- "Gann Gitter" auf angegebenen Koordinaten erstellen
    if(!ObjectCreate(chart_ID,name,OBJ_GANNGRID,sub_window,time1,price1,time2,0))
    {
        Print(__FUNCTION__,
              ": Objekt \" Gann Gitter\" konnte nicht erstellt werden! Fehlercode = ",GetLastError());
        return(false);
    }
    //--- Ändern wir den Maßstab (Anzahl der Pips pro Bar)
    ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
    //--- Ändern wir die Richtung des Trends von "Gann Gitter" (true - absteigend, falsch - aufsteigend)
    ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction);
    //--- Gitterfarbe setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
    //--- Gitterlinienstil angeben
    ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
    //--- Breite der Gitterlinien einstellen
    ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
    //--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
}

```

```

//--- aktivieren (true) oder deaktivieren (false) Wählen des Gitters für Bewegung
//--- Wenn Sie ein grafisches Objekt durch Funktion ObjectCreate erstellen, das Objekt
//--- kann nicht ausgewählt und verschoben werden. Innerhalb dieser Methode ist der St
//--- true, was Sie das Objekt auswählen und verschieben erlaubt
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der
    ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- setzen die Priorität für eine Mausklick-Ereignisse auf dem Chart
    ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
    return(true);
; }

//+-----+
//| Verschiebt den Ankerpunkt von Gann Gitter |
//+-----+
bool GannGridPointChange(const long   chart_ID=0,      // ID des Charts
                        const string name="GannGrid",  // Gittername
                        const int    point_index=0,   // Nummer des Ankerpunktes
                        datetime     time=0,          // Zeitkoordinate des Punktes
                        double        price=0)         // Preiskoordinate des Punktes
{
//--- Wenn die Koordinaten des Punktes nicht angegeben sind, dann verschieben wir es a
    if(!time)
        time=TimeCurrent();
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Gitterankerpunkt bewegen
    if(!ObjectMove(chart_ID,name,point_index,time,price))
    {
        Print(__FUNCTION__,
              ": der Bezugspunkt konnte nicht bewegt werden! Fehlercode = ",GetLastError());
        return(false);
    }
; }
//--- die erfolgreiche Umsetzung
    return(true);
; }

//+-----+
//| Ändert den Maßstab von "Gann Gitter" |
//+-----+
bool GannGridScaleChange(const long   chart_ID=0,      // ID des Charts
                        const string name="GannGrid",  // Gitter
                        const double scale=1.0)         // Maßstab
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Ändern wir den Maßstab (Anzahl der Pips pro Bar)

```

```

    if(!ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale))
    {
        Print(__FUNCTION__,
              ": Konnte nicht den Maßstab ändern! Fehlercode = ",GetLastError());
        return(false);
    }
; }

//--- die erfolgreiche Umsetzung
return(true);
; }

//+-----+
//| Ändert die Trendrichtung von "Gann Gitter" |
//+-----+
bool GannGridDirectionChange(const long   chart_ID=0,      // ID des Charts
                             const string name="GannGrid", // Gittername
                             const bool   direction=true)   // Trendrichtung
{
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Ändern wir die Trendrichtung von "Gann Gitter"
    if(!ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction))
    {
        Print(__FUNCTION__,
              ": Konnte nicht die Trendrichtung ändern! Fehlercode = ",GetLastError());
        return(false);
    }
; }

//--- die erfolgreiche Umsetzung
return(true);
; }

//+-----+
//| Die Funktion löscht das Objekt "Gann Gitter" |
//+-----+
bool GannGridDelete(const long   chart_ID=0,      // ID des Charts
                    const string name="GannGrid") // Gittername
{
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Gann Gitter löschen
    if(!ObjectDelete(chart_ID,name))
    {
        Print(__FUNCTION__,
              ": Objekt \" Gann Gitter\" konnte nicht gelöscht werden! Fehlercode = ",GetLastError());
        return(false);
    }
; }

//--- die erfolgreiche Umsetzung
return(true);
; }

//+-----+
//| Überprüft die Werte der Ankerpunkten von Gann Gitter |
//| und setzt Standardwerte für leere Werte |

```

```

//+-----+
void ChangeGannGridEmptyPoints(datetime &time1,double &price1,datetime &time2)
{
//--- wenn die Zeit des zweiten Punktes nicht angegeben ist, wird sie auf dem aktuellen
    if(!time2)
        time2=TimeCurrent();
//--- wenn die Zeit des ersten Punktes nicht angegeben ist, wird sie auf 9 Bars nach
    if(!time1)
    {
        //--- Array um Eröffnungszeit der letzten 10 Bars aufzunehmen
        datetime temp[10];
        CopyTime(Symbol(),Period(),time2,10,temp);
        //--- stellen wir den ersten Punkt 9 Bars links dem zweiten Punkt
        time1=temp[0];
    };
//--- Wenn der Preis des ersten Punktes nicht angegeben wird, dann wird es einen Wert
    if(!price1)
        price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
};
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//--- Überprüfen die Richtigkeit der Eingabeparameter
    if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
        InpDate2<0 || InpDate2>100)
    {
        Print("Fehler! Ungültige Eingabeparameter! ");
        return;
    };
//--- Die Anzahl der sichtbaren Bars im Chart-Fenster
    int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- Größe des Arrays price
    int accuracy=1000;
//--- Arrays für Werte von Daten und Preise, die für Setzung und Änderung
//--- der Koordinaten der Ankerpunkten des Gitters verwendet werden
    datetime date[];
    double price[];
//--- Speicher reservieren
    ArrayResize(date,bars);
    ArrayResize(price,accuracy);
//--- Füllen die Anordnung von Daten
    ResetLastError();
    if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
    {
        Print("Kann nicht den Wert der Zeit kopieren! Fehlercode = ",GetLastError());
        return;
    };
};

```

```

//--- Füllen den Array der Preise
//--- den Minimal- und Maximalwert der Charts finden
double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- Schritte der Preisänderung finden und das Array füllen
double step=(max_price-min_price)/accuracy;
for(int i=0;i<accuracy;i++)
    price[i]=min_price+i*step;
//--- Definieren die Punkte um Gann Gitter zu zeichnen
int d1=InpDate1*(bars-1)/100;
int d2=InpDate2*(bars-1)/100;
int p1=InpPrice1*(accuracy-1)/100;
//--- Gann Gitter erstellen
if(!GannGridCreate(0,InpName,0,date[d1],price[p1],date[d2],InpScale,InpDirection,
    InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
{
    return;
};
//--- den Chart neu zeichnen und 1 Sekunde warten
ChartRedraw();
Sleep(1000);
//--- jetzt bewegen wir die Gitterankerpunkte
//--- Schleifenzähler
int v_steps=accuracy/4;
//--- Wir bewegen den ersten Bezugspunkt vertikal
for(int i=0;i<v_steps;i++)
{
    //--- den nächsten Wert nehmen
    if(p1<accuracy-1)
        p1+=1;
    if(!GannGridPointChange(0,InpName,0,date[d1],price[p1]))
        return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
    if(IsStopped())
        return;
    //--- den Chart neu zeichnen
    ChartRedraw();
};
//--- 1 Sekunde Verzögerung
Sleep(1000);
//--- Schleifenzähler
int h_steps=bars/4;
//--- den zweiten Punkt horizontal bewegen
for(int i=0;i<h_steps;i++)
{
    //--- den nächsten Wert nehmen
    if(d2<bars-1)
        d2+=1;
    if(!GannGridPointChange(0,InpName,1,date[d2],0))

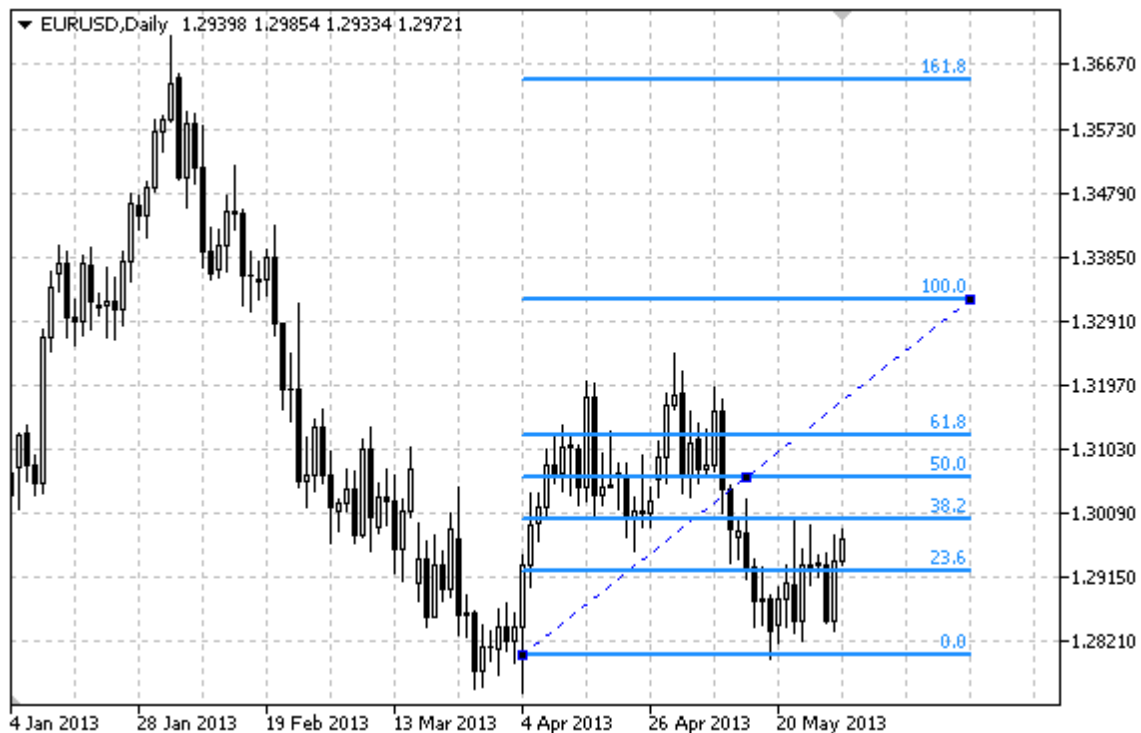
```

```
        return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
    if(IsStopped())
        return;
//--- den Chart neu zeichnen
    ChartRedraw();
// Verzögerung 0.05 Sekunden
    Sleep(50);
    ;}
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//--- Ändern wir die Trendrichtung des Gitters zu absteigend
    GannGridDirectionChange(0, InpName, true);
//--- den Chart neu zeichnen
    ChartRedraw();
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//--- das Gitter aus dem Chart löschen
    GannGridDelete(0, InpName);
    ChartRedraw();
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//---
    ;}
```



## OBJ\_FIBO

Fibonacci Levels.



### Hinweis

Für das Objekt "Fibonacci Levels" können Sie die weitere Anzeige nach rechts und/oder nach links (Eigenschaften [OBJPROP\\_RAY\\_RIGHT](#) und [OBJPROP\\_RAY\\_LEFT](#) jeweils) angeben.

Sie können auch die Anzahl der Linien-Ebenen, ihre Werte und Farben angeben.

### Beispiel

Das folgende Skript erstellt und bewegt auf dem Chart Objekt "Fibonacci Levels". Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Das Skript erstellt ein graphisches Objekt \"Fibonacci Levels\"
#property description "Koordinaten des Bezugspunktes werden in Prozent von"
#property description "der Größe des Chartfensters eingegeben."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="FiboLevels";           // Objektname
input int         InpDate1=10;                    // Datum des ersten Punktes in %
input int         InpPrice1=65;                   // Preis des ersten Punktes in %
input int         InpDate2=90;                    // Datum des zweiten Punktes in %
input int         InpPrice2=85;                   // Preis des zweiten Punktes in %
input color       InpColor=clrRed;                // Objektfarbe
```

```

input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Linienstil
input int              InpWidth=2;                // Linienbreite
input bool             InpBack=false;             // Objekt im Hintergrund
input bool             InpSelection=true;         // Wählen um zu bewegen
input bool             InpRayLeft=false;          // Fortsetzung des Objekts nach links
input bool             InpRayRight=false;         // Fortsetzung des Objekts nach rechts
input bool             InpHidden=true;            // Ausgeblendet in der Objektliste
input long             InpZOrder=0;               // Priorität auf Mausklick

//+-----+
//| Erstellt Objekt Fibonacci Levels auf angegebenen Koordinaten |
//+-----+

bool FiboLevelsCreate(const long      chart_ID=0,      // ID des Charts
                     const string    name="FiboLevels", // Objektname
                     const int       sub_window=0,    // Nummer des Unterfens
                     datetime         time1=0,        // Zeit des ersten Punk
                     double           price1=0,       // Preis des ersten Pur
                     datetime         time2=0,        // Zeit des zweiten Pur
                     double           price2=0,       // Preis des zweiten Pu
                     const color      clr=clrRed,     // Objektfarbe
                     const ENUM_LINE_STYLE style=STYLE_SOLID, // Linienstil des Objek
                     const int        width=1,        // Breite der Objektlin
                     const bool       back=false,     // Im Hintergrund
                     const bool       selection=true,  // Wählen um zu bewegen
                     const bool       ray_left=false, // Fortsetzung des Objek
                     const bool       ray_right=false, // Fortsetzung des Objek
                     const bool       hidden=true,    // Ausgeblendet in der
                     const long       z_order=0)      // Priorität auf Mausklick
{
    //--- Die Koordinaten der Bezugspunkte angeben, wenn sie nicht gesetzt sind
    ChangeFiboLevelsEmptyPoints(time1,price1,time2,price2);
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Fibonacci Levels auf angegebenen Koordinaten erstellen
    if(!ObjectCreate(chart_ID,name,OBJ_FIBO,sub_window,time1,price1,time2,price2))
    {
        Print(__FUNCTION__,
              ": Objekt \" Fibonacci Levels\" konnte nicht erstellt werden! Fehlercode =
        return(false);
    };
    //--- Farbe setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
    //--- Linienstil setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
    //--- Linienbreite setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
    //--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
    //--- aktivieren (true) oder deaktivieren (false) Bewegung des Objekts für Bewegung
    //--- Wenn Sie ein grafisches Objekt durch Funktion ObjectCreate erstellen, das Objekt

```

```

//--- kann nicht ausgewählt und verschoben werden. Innerhalb dieser Methode ist der St
//--- true, was Sie das Objekt auswählen und verschieben erlaubt
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- aktivieren (true) oder deaktivieren (false) Fortsetzung des Objekts nach links
    ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- aktivieren (true) oder deaktivieren (false) Fortsetzung des Objekts nach rechts
    ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der
    ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- setzen die Priorität für eine Mausklick-Ereignisse auf dem Chart
    ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
    return(true);
; }
//+-----+
//| Gibt die Anzahl der Ebenen und ihren Parameter an |
//+-----+
bool FiboLevelsSet(int          levels,          // Anzahl der Ebenelinien
                  double        &values[],      // Werte der Linien
                  color          &colors[],      // Farbe der Linien
                  ENUM_LINE_STYLE &styles[],     // Linienstil
                  int            &widths[],      // Breite der Linien
                  const long      chart_ID=0,    // ID des Charts
                  const string    name="FiboLevels") // Objektname
{
//--- Überprüfen wir die Größe des Arrays
    if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
        levels!=ArraySize(widths) || levels!=ArraySize(values))
    {
        Print(__FUNCTION__," : Länge des Arrays entspricht nicht der Anzahl der Ebenen, I
        return(false);
    }
; }
//--- Die Anzahl der Ebenen angeben
    ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- die Eigenschaften der Ebenen im Zyklus angeben
    for(int i=0;i<levels;i++)
    {
        //--- Wert der Ebene
        ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
        //--- Farbe der Ebene
        ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
        //--- Stil der Ebene
        ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
        //--- Stärke der Ebene
        ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
        //--- Beschreibung der Ebene
        ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
; }

```

```

//--- die erfolgreiche Umsetzung
    return(true);
; }
//+-----+
//| Verschiebt den Ankerpunkt von Fibonacci Levels |
//+-----+
bool FiboLevelsPointChange(const long   chart_ID=0,      // ID des Charts
                           const string name="FiboLevels", // Objektname
                           const int   point_index=0,    // Nummer des Punktes
                           datetime     time=0,          // Zeitkoordinate des Punktes
                           double       price=0)          // Preiskoordinate des Punktes
{
//--- Wenn die Koordinaten des Punktes nicht angegeben sind, dann verschieben wir es auf den aktuellen
    if(!time)
        time=TimeCurrent();
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- bewegen den Bezugspunkt
    if(!ObjectMove(chart_ID,name,point_index,time,price))
    {
        Print(__FUNCTION__,
              ": der Bezugspunkt konnte nicht bewegt werden! Fehlercode = ",GetLastError());
        return(false);
    }
; }
//--- die erfolgreiche Umsetzung
    return(true);
; }
//+-----+
//| Löscht das Objekt "Fibonacci Levels" |
//+-----+
bool FiboLevelsDelete(const long   chart_ID=0,      // ID des Charts
                      const string name="FiboLevels") // Objektname
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- das Objekt löschen
    if(!ObjectDelete(chart_ID,name))
    {
        Print(__FUNCTION__,
              ": Objekt \" Fibonacci Levels\" konnte nicht gelöscht werden! Fehlercode = ",GetLastError());
        return(false);
    }
; }
//--- die erfolgreiche Umsetzung
    return(true);
; }
//+-----+
//| Überprüft die Werte der Ankerpunkten von Fibonacci Levels und |

```

```

//| setzt Standardwerte für leere Werte |
//+-----+
void ChangeFiboLevelsEmptyPoints(datetime &time1,double &price1,
                                datetime &time2,double &price2)
{
//--- wenn die Zeit des zweiten Punktes nicht angegeben ist, wird sie auf dem aktuellen
    if(!time2)
        time2=TimeCurrent();
//--- wenn der Preis des zweiten Punktes nicht angegeben ist, wird er dem Bid-Wert gleich
    if(!price2)
        price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- wenn die Zeit des ersten Punktes nicht angegeben ist, wird sie auf 9 Baren nach
    if(!time1)
    {
        //--- Array um Eröffnungszeit der letzten 10 Bars aufzunehmen
        datetime temp[10];
        CopyTime(Symbol(),Period(),time2,10,temp);
        //--- stellen wir den ersten Punkt 9 Baren links dem zweiten Punkt
        time1=temp[0];
    };
//--- wenn der Preis des ersten Punktes nicht angegeben ist, bewegen wir ihn 200 Punkte
    if(!price1)
        price1=price2-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
};
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//--- Überprüfen die Richtigkeit der Eingabeparameter
    if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
        InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
    {
        Print("Fehler! Ungültige Eingabeparameter! ");
        return;
    };
//--- Die Anzahl der sichtbaren Bars im Chart-Fenster
    int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- Größe des Arrays price
    int accuracy=1000;
//--- Arrays für Werte von Daten und Preise, die für Setzung und Änderung
//--- der Koordinaten der Ankerpunkten von Fibonacci Levels verwendet werden
    datetime date[];
    double price[];
//--- Speicher reservieren
    ArrayResize(date,bars);
    ArrayResize(price,accuracy);
//--- Füllen die Anordnung von Daten
    ResetLastError();

```

```

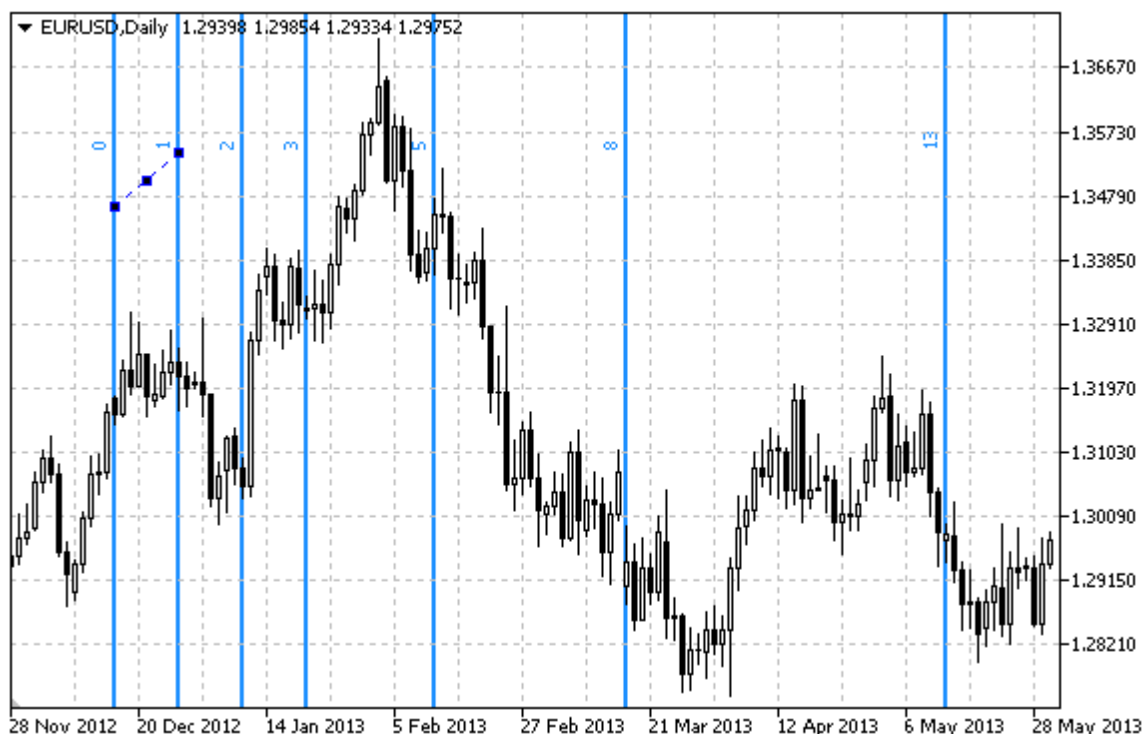
    if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
    {
        Print("Kann nicht den Wert der Zeit kopieren! Fehlercode = ",GetLastError());
        return;
    }
    ;}
//--- Füllen den Array der Preise
//--- den Minimal- und Maximalwert der Charts finden
    double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
    double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- Schritte der Preisänderung finden und das Array füllen
    double step=(max_price-min_price)/accuracy;
    for(int i=0;i<accuracy;i++)
        price[i]=min_price+i*step;
//--- Definieren die Punkte um Fibonacci Levels zu zeichnen
    int d1=InpDate1*(bars-1)/100;
    int d2=InpDate2*(bars-1)/100;
    int p1=InpPrice1*(accuracy-1)/100;
    int p2=InpPrice2*(accuracy-1)/100;
//--- Objekt erstellen
    if(!FiboLevelsCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor,
        InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder)
    {
        return;
    }
    ;}
//--- den Chart neu zeichnen und 1 Sekunde warten
    ChartRedraw();
    Sleep(1000);
//--- jetzt bewegen wir die Ankerpunkte
//--- Schleifenzähler
    int v_steps=accuracy*2/5;
//--- den ersten Punkt bewegen
    for(int i=0;i<v_steps;i++)
    {
        //--- den nächsten Wert nehmen
        if(p1>1)
            p1-=1;
        //--- den Punkt bewegen
        if(!FiboLevelsPointChange(0,InpName,0,date[d1],price[p1]))
            return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
        if(IsStopped())
            return;
        //--- den Chart neu zeichnen
        ChartRedraw();
    }
    ;}
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//--- Schleifenzähler
    v_steps=accuracy*4/5;

```

```
//--- den zweiten Punkt bewegen
for(int i=0;i<v_steps;i++)
{
    //--- den nächsten Wert nehmen
    if(p2>1)
        p2-=1;
    //--- den Punkt bewegen
    if(!FiboLevelsPointChange(0,InpName,1,date[d2],price[p2]))
        return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
if(IsStopped())
    return;
//--- den Chart neu zeichnen
ChartRedraw();
};
//--- 1 Sekunde Verzögerung
Sleep(1000);
//--- das Objekt aus dem Chart löschen
FiboLevelsDelete(0,InpName);
ChartRedraw();
//--- 1 Sekunde Verzögerung
Sleep(1000);
//---
};
```

## OBJ\_FIBOTIMES

Fibonacci Zeitzonen.



### Hinweis

Für "Fibonacci Zeitzonen" können Sie die Anzahl der Linien-Ebenen, ihre Werte und Farben angeben.

### Beispiel

Das folgende Skript erstellt und bewegt auf dem Chart Objekt "Fibonacci Zeitzonen". Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Das Skript erstellt ein graphisches Objekt \"Fibonacci Zeitzone\"
#property description "Koordinaten der Ankerpunkte werden in Pixeln angegeben"
#property description "Chartfenster."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="FiboTimes";           // Objektname
input int         InpDate1=10;                   // Datum des ersten Punktes in %
input int         InpPrice1=45;                   // Preis des ersten Punktes in %
input int         InpDate2=20;                   // Datum des zweiten Punktes in %
input int         InpPrice2=55;                   // Preis des zweiten Punktes in %
input color       InpColor=clrRed;                // Objektfarbe
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Linienstil
input int         InpWidth=2;                     // Linienbreite
input bool        InpBack=false;                  // Objekt im Hintergrund
```



```

input bool      InpSelection=true;           // Wählen um zu bewegen
input bool      InpHidden=true;             // Ausgeblendet in der Objektliste
input long      InpZOrder=0;                // Priorität auf Mausclick
//+-----+
//| Erstellt Objekt Fibonacci Zeitzonen auf angegebenen Koordinaten |
//+-----+
bool FiboTimesCreate(const long      chart_ID=0,           // ID des Charts
                    const string     name="FiboTimes",     // Objektname
                    const int        sub_window=0,        // Nummer des Unterfensters
                    datetime         time1=0,             // Zeit des ersten Punktes
                    double            price1=0,           // Preis des ersten Punktes
                    datetime         time2=0,             // Zeit des zweiten Punktes
                    double            price2=0,           // Preis des zweiten Punktes
                    const color       clr=clrRed,         // Objektfarbe
                    const ENUM_LINE_STYLE style=STYLE_SOLID, // Linienstil des Objektes
                    const int         width=1,           // Breite der Objektlinie
                    const bool        back=false,        // In der Hintergrund
                    const bool        selection=true,     // Wählen um zu bewegen
                    const bool        hidden=true,        // Verborgenen in der Liste der Objekte
                    const long        z_order=0)          // Priorität für Mausclick
{
//--- Die Koordinaten der Bezugspunkte angeben, wenn sie nicht gesetzt sind
    ChangeFiboTimesEmptyPoints(time1,price1,time2,price2);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- "Fibonacci Zeitzonen" auf angegebenen Koordinaten erstellen
    if(!ObjectCreate(chart_ID,name,OBJ_FIBOTIMES,sub_window,time1,price1,time2,price2))
    {
        Print(__FUNCTION__,
              ": Objekt \" Fibonacci Zeitzonen\" konnte nicht erstellt werden! Fehlercode: ",
              GetLastError(),
              "\n");
        return(false);
    }
//--- Farbe setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- Linienstil setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- Linienbreite setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- aktivieren (true) oder deaktivieren (false) Bewegung des Objekts für Bewegung
//--- Wenn Sie ein grafisches Objekt durch Funktion ObjectCreate erstellen, das Objekt
//--- kann nicht ausgewählt und verschoben werden. Innerhalb dieser Methode ist der Status
//--- true, was Sie das Objekt auswählen und verschieben erlaubt
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der Liste
    ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- setzen die Priorität für eine Mausclick-Ereignisse auf dem Chart

```

```

    ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Gibt die Anzahl der Ebenen und ihren Parameter an |
//+-----+
bool FiboTimesLevelsSet(int          levels,          // Anzahl der Ebenenlinien
                        double       &values[],       // Werte der Linien
                        color        &colors[],       // Farbe der Linien
                        ENUM_LINE_STYLE &styles[],     // Linienstil
                        int          &widths[],       // Linienbreite
                        const long    chart_ID=0,      // ID des Charts
                        const string  name="FiboTimes") // Objektname
{
//--- Überprüfen wir die Größe des Arrays
    if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
        levels!=ArraySize(widths) || levels!=ArraySize(widths))
    {
        Print(__FUNCTION__,": Länge des Arrays entspricht nicht der Anzahl der Ebenen, ");
        return(false);
    }
};
//--- Die Anzahl der Ebenen angeben
    ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- die Eigenschaften der Ebenen im Zyklus angeben
    for(int i=0;i<levels;i++)
    {
        //--- Wert der Ebene
        ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
        //--- Farbe der Ebene
        ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
        //--- Stil der Ebene
        ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
        //--- Stärke der Ebene
        ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
        //--- Beschreibung der Ebene
        ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(values[i],1));
    }
//--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Verschiebt den Ankerpunkt von Fibonacci Zeitzone |
//+-----+
bool FiboTimesPointChange(const long    chart_ID=0,      // ID des Charts
                          const string  name="FiboTimes", // Objektname
                          const int     point_index=0,   // Nummer des Ankerpunktes
                          datetime      time=0,         // Zeitkoordinate des Punktes
                          double        price=0)         // Preiskoordinate des Punktes

```

```

{
//--- Wenn die Koordinaten des Punktes nicht angegeben sind, dann verschieben wir es a
    if(!time)
        time=TimeCurrent();
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- bewegen den Bezugspunkt
    if(!ObjectMove(chart_ID,name,point_index,time,price))
    {
        Print(__FUNCTION__,
            ": der Bezugspunkt konnte nicht bewegt werden! Fehlercode = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Löscht das Objekt "Fibonacci Zeitzonen" |
//+-----+
bool FiboTimesDelete(const long   chart_ID=0,      // ID des Charts
                    const string name="FiboTimes") // Objektname
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- das Objekt löschen
    if(!ObjectDelete(chart_ID,name))
    {
        Print(__FUNCTION__,
            ": Objekt \" Fibonacci Zeitzonen\" konnte nicht gelöscht werden! Fehlercode = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Prüft die Werte der Ankerpunkten von "Fibonacci Zeitzonen" |
//| und setzt Standardwerte für leere Werte |
//+-----+
void ChangeFiboTimesEmptyPoints(datetime &time1,double &pricel,
                                datetime &time2,double &price2)
{
//--- Wenn Zeit des ersten Punktes nicht angegeben ist, wird es auf dem aktuellen Bar
    if(!time1)
        time1=TimeCurrent();
//--- Wenn der Preis des ersten Punktes nicht angegeben wird, dann wird es einen Wert
    if(!pricel)
        pricel=SymbolInfoDouble(Symbol(),SYMBOL_BID);

```

```

//--- wenn die Zeit des zweiten Punktes nicht angegeben ist, wird sie auf 2 Balken nach
if(!time2)
{
    //--- Array um die Zeit der Öffnung der letzten 3 Bars zu empfangen
    datetime temp[3];
    CopyTime(Symbol(),Period(),time1,3,temp);
    //--- stellen wir den ersten Punkt 2 Bars links auf dem zweiten Punkt
    time2=temp[0];
};
//--- Wenn der Preis des zweiten Punktes nicht angegeben wird, dann wird es der Preis
if(!price2)
    price2=price1;
};
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    //--- Überprüfen die Richtigkeit der Eingabeparameter
    if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
        InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
    {
        Print("Fehler! Ungültige Eingabeparameter! ");
        return;
    };
    //--- Die Anzahl der sichtbaren Bars im Chart-Fenster
    int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
    //--- Größe des Arrays price
    int accuracy=1000;
    //--- Arrays für Werte von Daten und Preise, die für Setzung und Änderung
    //--- der Koordinaten der Ankerpunkten von Fibonacci Zeitzone verwendet werden
    datetime date[];
    double price[];
    //--- Speicher reservieren
    ArrayResize(date,bars);
    ArrayResize(price,accuracy);
    //--- Füllen die Anordnung von Daten
    ResetLastError();
    if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
    {
        Print("Kann nicht den Wert der Zeit kopieren! Fehlercode = ",GetLastError());
        return;
    };
    //--- Füllen den Array der Preise
    //--- den Minimal- und Maximalwert der Charts finden
    double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
    double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
    //--- Schritte der Preisänderung finden und das Array füllen
    double step=(max_price-min_price)/accuracy;

```

```

    for(int i=0;i<accuracy;i++)
        price[i]=min_price+i*step;
//--- Definieren die Punkte um Fibonacci Zeitzonen zu zeichnen
    int d1=InpDate1*(bars-1)/100;
    int d2=InpDate2*(bars-1)/100;
    int p1=InpPrice1*(accuracy-1)/100;
    int p2=InpPrice2*(accuracy-1)/100;
//--- Objekt erstellen
    if(!FiboTimesCreate(0, InpName, 0, date[d1], price[p1], date[d2], price[p2],
        InpColor, InpStyle, InpWidth, InpBack, InpSelection, InpHidden, InpZOrder))
    {
        return;
    }
//--- den Chart neu zeichnen und 1 Sekunde warten
    ChartRedraw();
    Sleep(1000);
//--- jetzt bewegen wir die Ankerpunkte
//--- Schleifenzähler
    int h_steps=bars*2/5;
//--- den zweiten Punkt bewegen
    for(int i=0;i<h_steps;i++)
    {
        //--- den nächsten Wert nehmen
        if(d2<bars-1)
            d2+=1;
        //--- den Punkt bewegen
        if(!FiboTimesPointChange(0, InpName, 1, date[d2], price[p2]))
            return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
        if(IsStopped())
            return;
        //--- den Chart neu zeichnen
        ChartRedraw();
        // Verzögerung 0.05 Sekunden
        Sleep(50);
    }
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//--- Schleifenzähler
    h_steps=bars*3/5;
//--- den ersten Punkt bewegen
    for(int i=0;i<h_steps;i++)
    {
        //--- den nächsten Wert nehmen
        if(d1<bars-1)
            d1+=1;
        //--- den Punkt bewegen
        if(!FiboTimesPointChange(0, InpName, 0, date[d1], price[p1]))
            return;

```

```
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
    if(IsStopped())
        return;
    //--- den Chart neu zeichnen
    ChartRedraw();
    // Verzögerung 0.05 Sekunden
    Sleep(50);
; }
//--- 1 Sekunde Verzögerung
Sleep(1000);
//--- das Objekt aus dem Chart löschen
FiboTimesDelete(0, InpName);
ChartRedraw();
//--- 1 Sekunde Verzögerung
Sleep(1000);
//---
; }
```

## OBJ\_FIBOFAN

Fibonacci-Fan.



### Hinweis

Für "Fibonacci Fan" können Sie die Anzahl der Linien-Ebenen, ihre Werte und Farben angeben.

### Beispiel

Das folgende Skript erstellt und bewegt auf dem Chart Objekt "Fibonacci-Fan". Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Das Skript erstellt ein graphisches Objekt \"Fibonacci-Fan\"."
#property description "Koordinaten des Bezugspunktes werden in Prozent von"
#property description "der Größe des Chartfensters eingegeben."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="FiboFan";           // Name
input int         InpDate1=10;                 // Datum des ersten Punktes in %
input int         InpPrice1=25;                // Preis des ersten Punktes in %
input int         InpDate2=30;                 // Datum des zweiten Punktes in %
input int         InpPrice2=50;                // Preis des zweiten Punktes in %
input color       InpColor=clrRed;             // Linienfarbe
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Linienstil
input int         InpWidth=2;                  // Linienbreite
```

```

input bool      InpBack=false;           // Objekt im Hintergrund
input bool      InpSelection=true;        // Wählen um zu bewegen
input bool      InpHidden=true;          // Ausgeblendet in der Objektliste
input long      InpZOrder=0;             // Priorität auf Mausklick
//+-----+
//| Erstellt Objekt Fibonacci-Fan auf angegebenen Koordinaten |
//+-----+
bool FiboFanCreate(const long      chart_ID=0,          // ID des Charts
                  const string    name="FiboFan",      // Name von Fan
                  const int       sub_window=0,        // Nummer des Unterfensters
                  datetime        time1=0,             // Zeit des ersten Punktes
                  double          price1=0,            // Preis des ersten Punktes
                  datetime        time2=0,             // Zeit des zweiten Punktes
                  double          price2=0,            // Preis des zweiten Punktes
                  const color     clr=clrRed,          // Fan Linienfarbe
                  const ENUM_LINE_STYLE style=STYLE_SOLID, // Fan Linienstil
                  const int       width=1,             // Fan Linienbreite
                  const bool      back=false,          // Im Hintergrund
                  const bool      selection=true,       // Auswählen um zu bewegen
                  const bool      hidden=true,         // Ausgeblendet in der Obj
                  const long      z_order=0)           // Priorität auf Mausklick
{
//--- Die Koordinaten der Bezugspunkte angeben, wenn sie nicht gesetzt sind
    ChangeFiboFanEmptyPoints(time1,price1,time2,price2);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- "Fibonacci Fan" auf angegebenen Koordinaten erstellen
    if(!ObjectCreate(chart_ID,name,OBJ_FIBOFAN,sub_window,time1,price1,time2,price2))
    {
        Print(__FUNCTION__,
              ": Objekt \" Fibonacci Fan\" konnte nicht erstellt werden! Fehlercode = ",
              return(false);
    }
; }
//--- Farbe setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- Linienstil setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- Linienbreite setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- aktivieren (true) oder deaktivieren (false) Wählen von Fan für Bewegung
//--- Wenn Sie ein grafisches Objekt durch Funktion ObjectCreate erstellen, das Objekt
//--- kann nicht ausgewählt und verschoben werden. Innerhalb dieser Methode ist der St
//--- true, was Sie das Objekt auswählen und verschieben erlaubt
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der
    ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);

```



```

//--- setzen die Priorität für eine Mausklick-Ereignisse auf dem Chart
ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
return(true);
;
}
//+-----+
//| Gibt die Anzahl der Ebenen und ihren Parameter an |
//+-----+
bool FiboFanLevelsSet(int          levels,          // Anzahl der Ebenelinien
                     double        &values[],      // Werte der Linien
                     color          &colors[],      // Farbe der Ebenelinien
                     ENUM_LINE_STYLE &styles[],     // Stil der Ebenelinien
                     int            &widths[],      // Breite der Linien
                     const long     chart_ID=0,     // ID des Charts
                     const string   name="FiboFan") // Name von Fan
{
//--- Überprüfen wir die Größe des Arrays
if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
    levels!=ArraySize(widths) || levels!=ArraySize(widths))
{
    Print(__FUNCTION__,": Länge des Arrays entspricht nicht der Anzahl der Ebenen, ");
    return(false);
}
;
//--- Die Anzahl der Ebenen angeben
ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- die Eigenschaften der Ebenen im Zyklus angeben
for(int i=0;i<levels;i++)
{
    //--- Wert der Ebene
    ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
    //--- Farbe der Ebene
    ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
    //--- Stil der Ebene
    ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
    //--- Stärke der Ebene
    ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
    //--- Beschreibung der Ebene
    ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1));
}
;
//--- die erfolgreiche Umsetzung
return(true);
;
}
//+-----+
//| Verschiebt den Ankerpunkt von Fibonacci Fan |
//+-----+
bool FiboFanPointChange(const long     chart_ID=0,    // ID des Charts
                       const string   name="FiboFan", // Name von Fan
                       const int      point_index=0, // Nummer des Punktes
                       datetime        time=0,       // Zeitkoordinate des Punktes

```

```

        double price=0) // Preiskordinate des Punktes
    {
//--- Wenn die Koordinaten des Punktes nicht angegeben sind, dann verschieben wir es a
    if(!time)
        time=TimeCurrent();
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- bewegen den Bezugspunkt
    if(!ObjectMove(chart_ID,name,point_index,time,price))
    {
        Print(__FUNCTION__,
            ": der Bezugspunkt konnte nicht bewegt werden! Fehlercode = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
};

//+-----+
//| Löscht das Objekt "Fibonacci Fan" |
//+-----+
bool FiboFanDelete(const long chart_ID=0, // ID des Charts
                  const string name="FiboFan") // Name von Fan
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Fan löschen
    if(!ObjectDelete(chart_ID,name))
    {
        Print(__FUNCTION__,
            ": Objekt \" Fibonacci Fan\" konnte nicht gelöscht werden! Fehlercode = ",
            GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
};

//+-----+
//| Überprüft die Werte der Ankerpunkten von Fibonacci-Fan und |
//| setzt Standardwerte für leere Werte |
//+-----+
void ChangeFiboFanEmptyPoints(datetime &time1,double &price1,
                              datetime &time2,double &price2)
{
//--- wenn die Zeit des zweiten Punktes nicht angegeben ist, wird sie auf dem aktuelle
    if(!time2)
        time2=TimeCurrent();
//--- wenn der Preis des zweiten Punktes nicht angegeben ist, wird er dem Bid-Wert gle
    if(!price2)

```

```

    price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- wenn die Zeit des ersten Punktes nicht angegeben ist, wird sie auf 9 Baren nach
    if(!time1)
    {
        //--- Array um Eröffnungszeit der letzten 10 Bars aufzunehmen
        datetime temp[10];
        CopyTime(Symbol(),Period(),time2,10,temp);
        //--- stellen wir den ersten Punkt 9 Baren links dem zweiten Punkt
        time1=temp[0];
    };
//--- wenn der Preis des ersten Punktes nicht angegeben ist, bewegen wir ihn 200 Punkt
    if(!price1)
        price1=price2-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
};
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//--- Überprüfen die Richtigkeit der Eingabeparameter
    if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
        InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
    {
        Print("Fehler! Ungültige Eingabeparameter! ");
        return;
    };
//--- Die Anzahl der sichtbaren Bars im Chart-Fenster
    int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- Größe des Arrays price
    int accuracy=1000;
//--- Arrays für Werte von Daten und Preise, die für Setzung und Änderung
//--- der Koordinaten der Ankerpunkten von Fibonacci-Fan verwendet werden
    datetime date[];
    double price[];
//--- Speicher reservieren
    ArrayResize(date,bars);
    ArrayResize(price,accuracy);
//--- Füllen die Anordnung von Daten
    ResetLastError();
    if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
    {
        Print("Kann nicht den Wert der Zeit kopieren! Fehlercode = ",GetLastError());
        return;
    };
//--- Füllen den Array der Preise
//--- den Minimal- und Maximalwert der Charts finden
    double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
    double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- Schritte der Preisänderung finden und das Array füllen

```

```

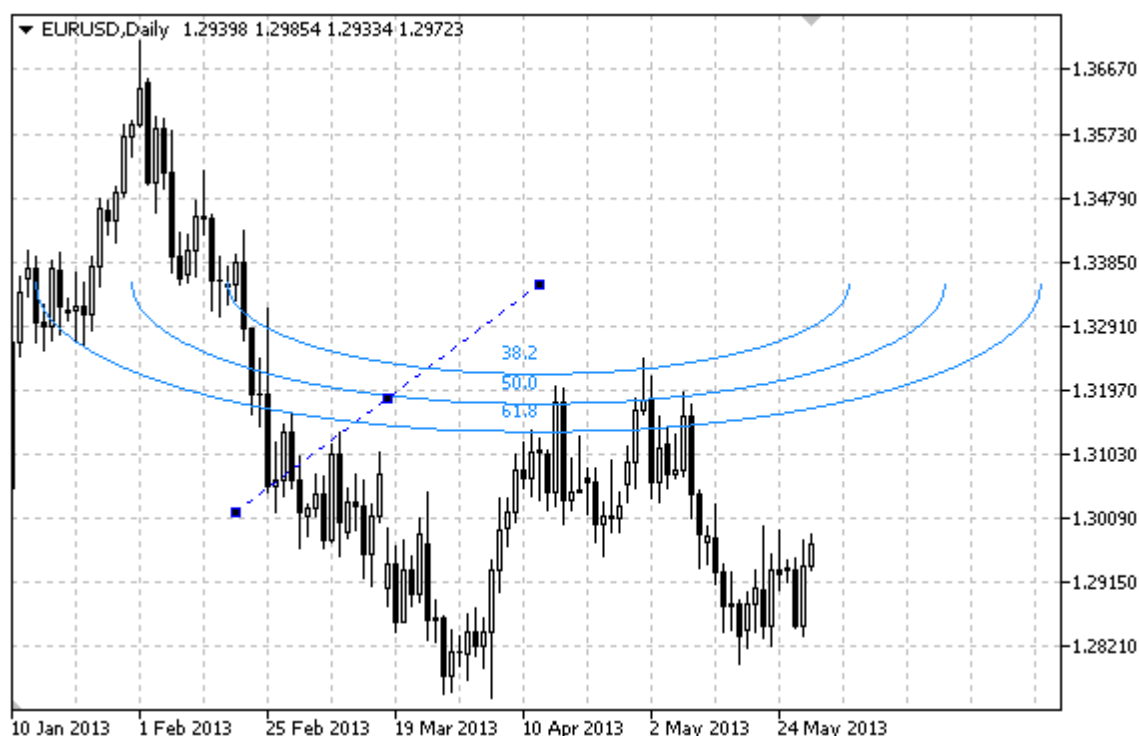
double step=(max_price-min_price)/accuracy;
for(int i=0;i<accuracy;i++)
    price[i]=min_price+i*step;
//--- Definieren die Punkte um Fibonacci-Fan zu zeichnen
int d1=InpDate1*(bars-1)/100;
int d2=InpDate2*(bars-1)/100;
int p1=InpPrice1*(accuracy-1)/100;
int p2=InpPrice2*(accuracy-1)/100;
//--- Objekt erstellen
if(!FiboFanCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],
    InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
{
    return;
};
//--- den Chart neu zeichnen und 1 Sekunde warten
ChartRedraw();
Sleep(1000);
//--- jetzt bewegen wir die Ankerpunkte von Fan
//--- Schleifenzähler
int v_steps=accuracy/2;
//--- den ersten Punkt bewegen
for(int i=0;i<v_steps;i++)
{
    //--- den nächsten Wert nehmen
    if(p1<accuracy-1)
        p1+=1;
    //--- den Punkt bewegen
    if(!FiboFanPointChange(0,InpName,0,date[d1],price[p1]))
        return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
    if(IsStopped())
        return;
    //--- den Chart neu zeichnen
    ChartRedraw();
};
//--- 1 Sekunde Verzögerung
Sleep(1000);
//--- Schleifenzähler
int h_steps=bars/4;
//--- den zweiten Punkt bewegen
for(int i=0;i<h_steps;i++)
{
    //--- den nächsten Wert nehmen
    if(d2<bars-1)
        d2+=1;
    //--- den Punkt bewegen
    if(!FiboFanPointChange(0,InpName,1,date[d2],price[p2]))
        return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript

```

```
    if(IsStopped())
        return;
    //--- den Chart neu zeichnen
    ChartRedraw();
    // Verzögerung 0.05 Sekunden
    Sleep(50);
; }
//--- 1 Sekunde Verzögerung
Sleep(1000);
//--- das Objekt aus dem Chart löschen
FiboFanDelete(0, InpName);
ChartRedraw();
//--- 1 Sekunde Verzögerung
Sleep(1000);
//---
; }
```

## OBJ\_FIBOARC

Fibonacci Arcs.



### Hinweis

Für "Fibonacci Arcs" können Sie Anzeigemodus der gesamten Ellipse angeben. Der Krümmungsradius der Linien kann durch Änderung des Maßstabs und Koordinaten der Ankerpunkten gesetzt werden.

Sie können auch die Anzahl der Linien-Ebenen, ihre Werte und Farben angeben.

### Beispiel

Das folgende Skript erstellt und bewegt auf dem Chart Objekt "Fibonacci Arcs". Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Das Skript erstellt ein graphisches Objekt \"Fibonacci Arcs\"."
#property description "Koordinaten des Bezugspunktes werden in Prozent von"
#property description "der Größe des Chartfensters eingegeben."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="FiboArc";           // Objektname
input int         InpDate1=25;                 // Datum des ersten Punktes in %
input int         InpPrice1=25;                // Preis des ersten Punktes in %
input int         InpDate2=35;                 // Datum des zweiten Punktes in %
input int         InpPrice2=55;                // Preis des zweiten Punktes in %
input double      InpScale=3.0;                // Maßstab
```

```

input bool      InpFullEllipse=true;      // Form von Bögen
input color     InpColor=clrRed;          // Linienfarbe
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Linienstil
input int       InpWidth=2;               // Linienbreite
input bool      InpBack=false;            // Objekt im Hintergrund
input bool      InpSelection=true;        // Wählen um zu bewegen
input bool      InpHidden=true;           // Ausgeblendet in der Objektliste
input long      InpZOrder=0;              // Priorität auf Mausklick

//+-----+
//| Erstellt Objekt Fibonacci Arcs auf angegebenen Koordinaten |
//+-----+

bool FiboArcCreate(const long      chart_ID=0,      // ID des Charts
                  const string    name="FiboArc",  // Objektname
                  const int       sub_window=0,    // Nummer des Unterfensters
                  datetime         time1=0,        // Zeit des ersten Punktes
                  double          price1=0,        // Preis des ersten Punktes
                  datetime         time2=0,        // Zeit des zweiten Punktes
                  double          price2=0,        // Preis des zweiten Punktes
                  const double     scale=1.0,      // Maßstab
                  const bool       full_ellipse=false, // Form der Bögen
                  const color     clr=clrRed,     // Linienfarbe
                  const ENUM_LINE_STYLE style=STYLE_SOLID, // Linienstil
                  const int       width=1,        // Linienbreite
                  const bool       back=false,    // Im Hintergrund
                  const bool       selection=true, // Wählen um zu bewegen
                  const bool       hidden=true,   // Ausgeblendet in der Objektliste
                  const long       z_order=0)     // Priorität auf Mausklick
{
    //--- Die Koordinaten der Bezugspunkte angeben, wenn sie nicht gesetzt sind
    ChangeFiboArcEmptyPoints(time1,price1,time2,price2);
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- "Fibonacci Arcs" auf angegebenen Koordinaten erstellen
    if(!ObjectCreate(chart_ID,name,OBJ_FIBOARC,sub_window,time1,price1,time2,price2))
    {
        Print(__FUNCTION__,
              ": Objekt \" Fibonacci Arcs\" konnte nicht erstellt werden! Fehlercode = ",
              GetLastError());
        return(false);
    }
    //--- die Skala setzen
    ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
    //--- Anzeige der Bögen als volle Ellipse (true) oder Hälfte (false)
    ObjectSetInteger(chart_ID,name,OBJPROP_ELLIPSE,full_ellipse);
    //--- Farbe setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
    //--- Linienstil setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
    //--- Linienbreite setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);

```

```

//--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- aktivieren (true) oder deaktivieren (false) Wählen der Bögen für Bewegung
//--- Wenn Sie ein grafisches Objekt durch Funktion ObjectCreate erstellen, das Objekt
//--- kann nicht ausgewählt und verschoben werden. Innerhalb dieser Methode ist der St
//--- true, was Sie das Objekt auswählen und verschieben erlaubt
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der
    ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- setzen die Priorität für eine Mausklick-Ereignisse auf dem Chart
    ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
    return(true);
; }
//+-----+
//| Gibt die Anzahl der Ebenen und ihren Parameter an |
//+-----+
bool FiboArcLevelsSet(int          levels,          // Anzahl der Ebenelinien
                     double       &values[],       // Werte der Linien
                     color        &colors[],       // Farbe der Ebenelinien
                     ENUM_LINE_STYLE &styles[],    // Stil der Ebenelinien
                     int          &widths[],       // Breite der Linien
                     const long   chart_ID=0,      // ID des Charts
                     const string name="FiboArc") // Objektname
{
//--- Überprüfen wir die Größe des Arrays
    if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
        levels!=ArraySize(widths) || levels!=ArraySize(values))
    {
        Print(__FUNCTION__," : Länge des Arrays entspricht nicht der Anzahl der Ebenen, I
        return(false);
    }
; }
//--- Die Anzahl der Ebenen angeben
    ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- die Eigenschaften der Ebenen im Zyklus angeben
    for(int i=0;i<levels;i++)
    {
        //--- Wert der Ebene
        ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
        //--- Farbe der Ebene
        ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
        //--- Stil der Ebene
        ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
        //--- Stärke der Ebene
        ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
        //--- Beschreibung der Ebene
        ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
; }

```



```

//--- die erfolgreiche Umsetzung
    return(true);
; }
//+-----+
//| Verschiebt den Ankerpunkt von Fibonacci Arcs |
//+-----+
bool FiboArcPointChange(const long   chart_ID=0,      // ID des Charts
                        const string name="FiboArc",  // Objektname
                        const int    point_index=0,   // Nummer des Punktes
                        datetime     time=0,          // Zeitkoordinate des Punktes
                        double        price=0)        // Preiskoordinate des Punktes
{
//--- Wenn die Koordinaten des Punktes nicht angegeben sind, dann verschieben wir es
    if(!time)
        time=TimeCurrent();
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- bewegen den Bezugspunkt
    if(!ObjectMove(chart_ID,name,point_index,time,price))
    {
        Print(__FUNCTION__,
              ": der Bezugspunkt konnte nicht bewegt werden! Fehlercode = ",GetLastError());
        return(false);
    }
; }
//--- die erfolgreiche Umsetzung
    return(true);
; }
//+-----+
//| Löscht das Objekt "Fibonacci Arcs" |
//+-----+
bool FiboArcDelete(const long   chart_ID=0,      // ID des Charts
                   const string name="FiboArc") // Objektname
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- das Objekt löschen
    if(!ObjectDelete(chart_ID,name))
    {
        Print(__FUNCTION__,
              ": Objekt \" Fibonacci Arcs\" konnte nicht gelöscht werden! Fehlercode = ");
        return(false);
    }
; }
//--- die erfolgreiche Umsetzung
    return(true);
; }
//+-----+
//| Überprüft die Werte der Ankerpunkten von Fibonacci Arcs |

```

```

//| und setzt Standardwerte für leere Werte |
//+-----+
void ChangeFiboArcEmptyPoints(datetime &time1,double &pricel,
                             datetime &time2,double &price2)
{
//--- wenn die Zeit des zweiten Punktes nicht angegeben ist, wird sie auf dem aktuellen
    if(!time2)
        time2=TimeCurrent();
//--- wenn der Preis des zweiten Punktes nicht angegeben ist, wird er dem Bid-Wert gle
    if(!price2)
        price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- wenn die Zeit des ersten Punktes nicht angegeben ist, wird sie auf 9 Baren nach
    if(!time1)
    {
        //--- Array um Eröffnungszeit der letzten 10 Bars aufzunehmen
        datetime temp[10];
        CopyTime(Symbol(),Period(),time2,10,temp);
        //--- stellen wir den ersten Punkt 9 Baren links dem zweiten Punkt
        time1=temp[0];
    };
//--- wenn der Preis des ersten Punktes nicht angegeben ist, bewegen wir ihn 300 Punkt
    if(!pricel)
        pricel=price2-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
};
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//--- Überprüfen die Richtigkeit der Eingabeparameter
    if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
        InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
    {
        Print("Fehler! Ungültige Eingabeparameter! ");
        return;
    };
//--- Die Anzahl der sichtbaren Bars im Chart-Fenster
    int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- Größe des Arrays price
    int accuracy=1000;
//--- Arrays für Werte von Daten und Preise, die für Setzung und Änderung
//--- der Koordinaten der Ankerpunkten von Fibonacci Arcs verwendet werden
    datetime date[];
    double price[];
//--- Speicher reservieren
    ArrayResize(date,bars);
    ArrayResize(price,accuracy);
//--- Füllen die Anordnung von Daten
    ResetLastError();

```

```

    if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
    {
        Print("Kann nicht den Wert der Zeit kopieren! Fehlercode = ",GetLastError());
        return;
    }
    ;}
//--- Füllen den Array der Preise
//--- den Minimal- und Maximalwert der Charts finden
    double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
    double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- Schritte der Preisänderung finden und das Array füllen
    double step=(max_price-min_price)/accuracy;
    for(int i=0;i<accuracy;i++)
        price[i]=min_price+i*step;
//--- Definieren die Punkte um Fibonacci Arcs zu zeichnen
    int d1=InpDate1*(bars-1)/100;
    int d2=InpDate2*(bars-1)/100;
    int p1=InpPrice1*(accuracy-1)/100;
    int p2=InpPrice2*(accuracy-1)/100;
//--- Objekt erstellen
    if(!FiboArcCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpScale,
        InpFullEllipse,InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrd
    {
        return;
    }
    ;}
//--- den Chart neu zeichnen und 1 Sekunde warten
    ChartRedraw();
    Sleep(1000);
//--- jetzt bewegen wir die Ankerpunkte
//--- Schleifenzähler
    int v_steps=accuracy/5;
//--- den ersten Punkt bewegen
    for(int i=0;i<v_steps;i++)
    {
        //--- den nächsten Wert nehmen
        if(p1<accuracy-1)
            p1+=1;
        //--- den Punkt bewegen
        if(!FiboArcPointChange(0,InpName,0,date[d1],price[p1]))
            return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
        if(IsStopped())
            return;
        //--- den Chart neu zeichnen
        ChartRedraw();
    }
    ;}
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//--- Schleifenzähler
    int h_steps=bars/5;

```

```
//--- den zweiten Punkt bewegen
for(int i=0;i<h_steps;i++)
{
    //--- den nächsten Wert nehmen
    if(d2<bars-1)
        d2+=1;
    //--- den Punkt bewegen
    if(!FiboArcPointChange(0,InpName,1,date[d2],price[p2]))
        return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
if(IsStopped())
    return;
//--- den Chart neu zeichnen
ChartRedraw();
// Verzögerung 0.05 Sekunden
Sleep(50);
};
//--- 1 Sekunde Verzögerung
Sleep(1000);
//--- das Objekt aus dem Chart löschen
FiboArcDelete(0,InpName);
ChartRedraw();
//--- 1 Sekunde Verzögerung
Sleep(1000);
//---
};
```

## OBJ\_FIBOCHANNEL

Fibonacci-Kanal.



### Hinweis

Für das Objekt "Fibonacci-Kanal" können Sie die weitere Anzeige nach rechts und/oder nach links (Eigenschaften [OBJPROP\\_RAY\\_RIGHT](#) und [OBJPROP\\_RAY\\_LEFT](#) jeweils) angeben.

Sie können auch die Anzahl der Linien-Ebenen, ihre Werte und Farben angeben.

### Beispiel

Das folgende Skript erstellt und bewegt auf dem Chart Objekt "Fibonacci-Kanal". Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Das Skript erstellt ein graphisches Objekt \"Fibonacci-Kanal\"."
#property description "Koordinaten des Bezugspunktes werden in Prozent von"
#property description "der Größe des Chartfensters eingegeben."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="FiboChannel";      // Kanalname
input int         InpDate1=20;                // Datum des ersten Punktes in %
input int         InpPrice1=10;               // Preis des ersten Punktes in %
input int         InpDate2=60;                // Datum des zweiten Punktes in %
input int         InpPrice2=30;               // Preis des zweiten Punktes in %
input int         InpDate3=20;                // Datum des dritten Punktes in %
```

```

input int      InpPrice3=25;           // Preis des dritten Punktes in %
input color    InpColor=clrRed;        // Kanalfarbe
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Stil der Kanallinien
input int      InpWidth=2;             // Kanallinienbreite
input bool     InpBack=false;          // Im Hintergrund
input bool     InpSelection=true;       // Wählen um zu bewegen
input bool     InpRayLeft=false;       // Fortsetzung des Kanals nach links
input bool     InpRayRight=false;      // Fortsetzung des Kanals nach rechts
input bool     InpHidden=true;         // Ausgeblendet in der Objektliste
input long     InpZOrder=0;            // Priorität auf Mausklick

//+-----+
//| Erstellt Objekt Fibonacci-Kanal auf angegebenen Koordinaten |
//+-----+

bool FiboChannelCreate(const long      chart_ID=0,           // ID des Charts
                      const string    name="FiboChannel",    // Kanalname
                      const int       sub_window=0,         // Nummer des Unterfensters
                      datetime        time1=0,              // Zeit des ersten Punktes
                      double          price1=0,             // Preis des ersten Punktes
                      datetime        time2=0,              // Zeit des zweiten Punktes
                      double          price2=0,             // Preis des zweiten Punktes
                      datetime        time3=0,              // Zeit des dritten Punktes
                      double          price3=0,             // Preis des dritten Punktes
                      const color     clr=clrRed,           // Kanalfarbe
                      const ENUM_LINE_STYLE style=STYLE_SOLID, // Stil der Kanallinien
                      const int       width=1,             // Breite der Kanallinien
                      const bool      back=false,          // Im Hintergrund
                      const bool      selection=true,       // Wählen zu bewegen
                      const bool      ray_left=false,       // Fortsetzung des Kanals nach links
                      const bool      ray_right=false,      // Fortsetzung des Kanals nach rechts
                      const bool      hidden=true,          // Verborgenen in der Objektliste
                      const long      z_order=0)            // Priorität für Mausereignisse
{
    //--- Die Koordinaten der Bezugspunkte angeben, wenn sie nicht gesetzt sind
    ChangeFiboChannelEmptyPoints(time1,price1,time2,price2,time3,price3);
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Kanal auf angegebenen Koordinaten erstellen
    if(!ObjectCreate(chart_ID,name,OBJ_FIBOCHANNEL,sub_window,time1,price1,time2,price2,time3,price3))
    {
        Print(__FUNCTION__,
              ": Objekt \" Fibonacci-Kanal\" konnte nicht erstellt werden! Fehlercode = ",
              GetLastError(),
              "\n");
        return(false);
    }
    //--- Kanalfarbe setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
    //--- Stil der Kanallinien einstellen
    ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
    //--- Breite der Kanallinien einstellen
    ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
}

```

```

//--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- aktivieren (true) oder deaktivieren (false) Wählen des Kanals für Bewegung
//--- Wenn Sie ein grafisches Objekt durch Funktion ObjectCreate erstellen, das Objekt
//--- kann nicht ausgewählt und verschoben werden. Innerhalb dieser Methode ist der St
//--- true, was Sie das Objekt auswählen und verschieben erlaubt
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- aktivieren (true) oder deaktivieren (false) Fortsetzung des Kanals nach links
    ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- aktivieren (true) oder deaktivieren (false) Fortsetzung des Kanals nach rechts
    ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der
    ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- setzen die Priorität für eine Mausklick-Ereignisse auf dem Chart
    ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
    return(true);
; }
//+-----+
//| Gibt die Anzahl der Ebenen und ihren Parameter an |
//+-----+
bool FiboChannelLevelsSet(int          levels,          // Anzahl der Ebenenlinien
                        double        &values[],        // Werte der Linien
                        color          &colors[],        // Farbe der Linien
                        ENUM_LINE_STYLE &styles[],        // Stil der Linien
                        int            &widths[],        // Linienbreite
                        const long     chart_ID=0,        // ID des Charts
                        const string    name="FiboChannel") // Objektname
{
//--- Überprüfen wir die Größe des Arrays
if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
    levels!=ArraySize(widths) || levels!=ArraySize(values))
{
    Print(__FUNCTION__," : Länge des Arrays entspricht nicht der Anzahl der Ebenen, F
    return(false);
; }
//--- Die Anzahl der Ebenen angeben
    ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- die Eigenschaften der Ebenen im Zyklus angeben
    for(int i=0;i<levels;i++)
    {
        //--- Wert der Ebene
        ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
        //--- Farbe der Ebene
        ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
        //--- Stil der Ebene
        ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
        //--- Stärke der Ebene

```

```

        ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
        //--- Beschreibung der Ebene
        ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
    };
//--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Verschiebt den Ankerpunkt von Fibonacci-Kanal |
//+-----+
bool FiboChannelPointChange(const long   chart_ID=0,           // ID des Charts
                           const string name="FiboChannel",   // Kanalname
                           const int    point_index=0,        // Nummer des Bezugspunkt
                           datetime      time=0,              // Zeitkoordinate des Bez
                           double        price=0)              // Preiskoordinate des Be
{
//--- Wenn die Koordinaten des Punktes nicht angegeben sind, dann verschieben wir es a
    if(!time)
        time=TimeCurrent();
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- bewegen den Bezugspunkt
    if(!ObjectMove(chart_ID,name,point_index,time,price))
    {
        Print(__FUNCTION__,
              ": der Bezugspunkt konnte nicht bewegt werden! Fehlercode = ",GetLastError
        return(false);
    };
//--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Löscht den Kanal |
//+-----+
bool FiboChannelDelete(const long   chart_ID=0,           // ID des Charts
                      const string name="FiboChannel") // Kanalname
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- den Kanal löschen
    if(!ObjectDelete(chart_ID,name))
    {
        Print(__FUNCTION__,
              ": Objekt \" Fibonacci-Kanal\" konnte nicht gelöscht werden! Fehlercode =
        return(false);
    };
//--- die erfolgreiche Umsetzung

```



```

    return(true);
};

//+-----+
//| Überprüft die Werte der Ankerpunkten von Fibonacci-Kanal und |
//| setzt Standardwerte für leere Werte                          |
//+-----+
void ChangeFiboChannelEmptyPoints(datetime &time1,double &price1,datetime &time2,
                                   double &price2,datetime &time3,double &price3)
{
    //--- wenn die Zeit des zweiten (rechten) Punktes nicht angegeben ist, wird sie auf de
    if(!time2)
        time2=TimeCurrent();
    //--- wenn der Preis des zweiten Punktes nicht angegeben ist, wird er dem Bid-Wert gle
    if(!price2)
        price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
    //--- wenn die Zeit des ersten (linken) Punktes nicht angegeben ist, wird sie auf 9 Ba
    if(!time1)
    {
        //--- Array um Eröffnungszeit der letzten 10 Bars aufzunehmen
        datetime temp[10];
        CopyTime(Symbol(),Period(),time2,10,temp);
        //--- stellen wir den ersten Punkt 9 Baren links dem zweiten Punkt
        time1=temp[0];
    }
    //--- wenn der Preis des ersten Punktes nicht angegeben ist, bewegen wir ihn 300 Punkt
    if(!price1)
        price1=price2+300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
    //--- wenn die Zeit des dritten Punktes nicht angegeben ist, wird sie der Zeit des zwe
    if(!time3)
        time3=time1;
    //--- wenn der Preis des dritten Punktes nicht angegeben ist, wird er dem Preis des zw
    if(!price3)
        price3=price2;
};

//+-----+
//| Script program start function                                |
//+-----+
void OnStart()
{
    //--- Überprüfen die Richtigkeit der Eingabeparameter
    if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
        InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
        InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
    {
        Print("Fehler! Ungültige Eingabeparameter! ");
        return;
    }
};

//--- Die Anzahl der sichtbaren Bars im Chart-Fenster
int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);

```

```

//--- Größe des Arrays price
    int accuracy=1000;
//--- Arrays für Werte von Daten und Preise, die für Setzung und Änderung
//--- der Koordinaten der Kanalbindungspunkten verwendet werden
    datetime date[];
    double price[];
//--- Speicher reservieren
    ArrayResize(date,bars);
    ArrayResize(price,accuracy);
//--- Füllen die Anordnung von Daten
    ResetLastError();
    if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
    {
        Print("Kann nicht den Wert der Zeit kopieren! Fehlercode = ",GetLastError());
        return;
    }
//--- Füllen den Array der Preise
//--- den Minimal- und Maximalwert der Charts finden
    double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
    double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- Schritte der Preisänderung finden und das Array füllen
    double step=(max_price-min_price)/accuracy;
    for(int i=0;i<accuracy;i++)
        price[i]=min_price+i*step;
//--- Definieren die Punkte um den Kanal zu zeichnen
    int d1=InpDate1*(bars-1)/100;
    int d2=InpDate2*(bars-1)/100;
    int d3=InpDate3*(bars-1)/100;
    int p1=InpPrice1*(accuracy-1)/100;
    int p2=InpPrice2*(accuracy-1)/100;
    int p3=InpPrice3*(accuracy-1)/100;
//--- Fibonacci-Kanal erstellen
    if(!FiboChannelCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],p
        InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidde
    {
        return;
    }
//--- den Chart neu zeichnen und 1 Sekunde warten
    ChartRedraw();
    Sleep(1000);
//--- jetzt bewegen wir die Kanalbindungspunkte
//--- Schleifenzähler
    int h_steps=bars/10;
//--- den ersten Punkt bewegen
    for(int i=0;i<h_steps;i++)
    {
        //--- den nächsten Wert nehmen
        if(d1>1)
            d1-=1;

```

```

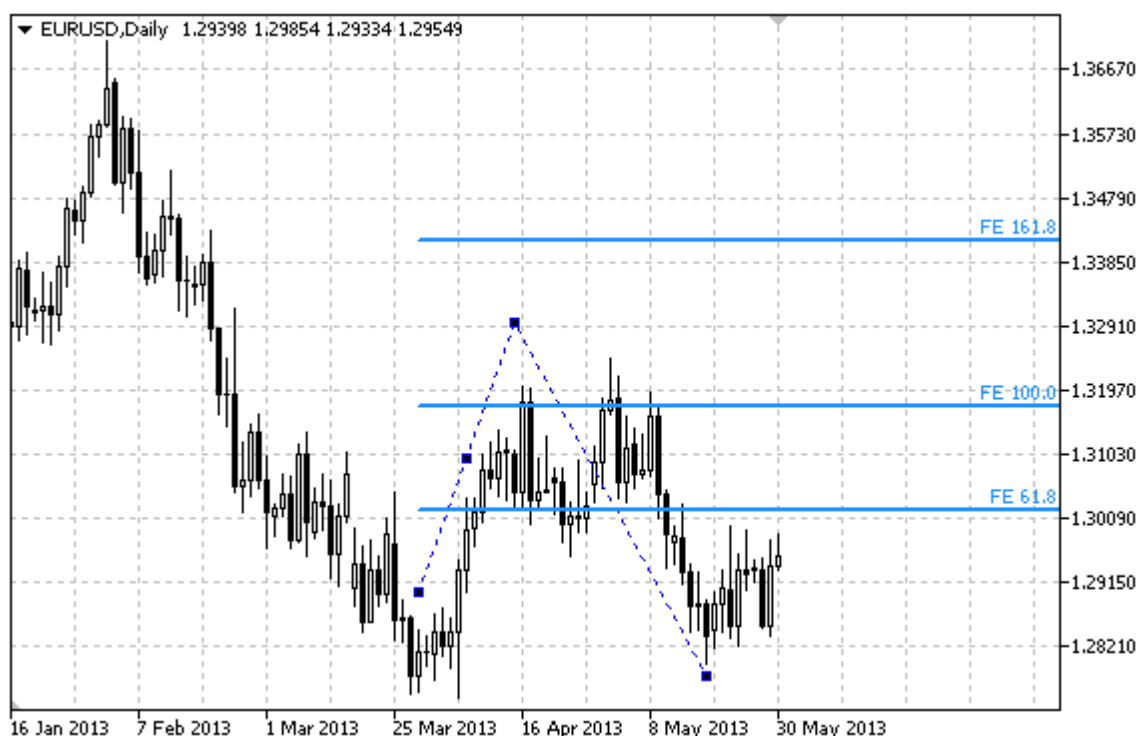
    //--- den Punkt bewegen
    if(!FiboChannelPointChange(0, InpName, 0, date[d1], price[p1]))
        return;
    //--- Überprüfen die Fakten von Zwangsabschaltung der Skript
    if(IsStopped())
        return;
    //--- den Chart neu zeichnen
    ChartRedraw();
    // Verzögerung 0.05 Sekunden
    Sleep(50);
    ;}
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//--- Schleifenzähler
    int v_steps=accuracy/10;
//--- den zweiten Punkt bewegen
    for(int i=0; i<v_steps; i++)
    {
        //--- den nächsten Wert nehmen
        if(p2>1)
            p2-=1;
        //--- den Punkt bewegen
        if(!FiboChannelPointChange(0, InpName, 1, date[d2], price[p2]))
            return;
        //--- Überprüfen die Fakten von Zwangsabschaltung der Skript
        if(IsStopped())
            return;
        //--- den Chart neu zeichnen
        ChartRedraw();
    }
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//--- Schleifenzähler
    v_steps=accuracy/15;
//--- den dritten Punkt bewegen
    for(int i=0; i<v_steps; i++)
    {
        //--- den nächsten Wert nehmen
        if(p3<accuracy-1)
            p3+=1;
        //--- den Punkt bewegen
        if(!FiboChannelPointChange(0, InpName, 2, date[d3], price[p3]))
            return;
        //--- Überprüfen die Fakten von Zwangsabschaltung der Skript
        if(IsStopped())
            return;
        //--- den Chart neu zeichnen
        ChartRedraw();
    }
    ;}

```

```
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//--- den Kanal aus dem Chart löschen
    FiboChannelDelete(0, InpName);
    ChartRedraw();
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//---
; }
```

## OBJ\_EXPANSION

Fibonacci Expansion.



### Hinweis

Für das Objekt "Fibonacci Expansion" können Sie die weitere Anzeige nach rechts und/oder nach links (Eigenschaften [OBJPROP\\_RAY\\_RIGHT](#) und [OBJPROP\\_RAY\\_LEFT](#) jeweils) angeben.

Sie können auch die Anzahl der Linien-Ebenen, ihre Werte und Farben angeben.

### Beispiel

Das folgende Skript erstellt und bewegt auf dem Chart Objekt "Fibonacci Expansion". Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Das Skript erstellt ein graphisches Objekt \"Fibonacci Expansion\"
#property description "Koordinaten des Bezugspunktes werden in Prozent von \"
#property description "der Größe des Chartfensters eingegeben."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="FiboExpansion";    // Objektname
input int         InpDate1=10;                // Datum des ersten Punktes in %
input int         InpPrice1=55;               // Preis des ersten Punktes in %
input int         InpDate2=30;               // Datum des zweiten Punktes in %
input int         InpPrice2=10;              // Preis des zweiten Punktes in %
input int         InpDate3=80;               // Datum des dritten Punktes in %
```

```

input int      InpPrice3=75;           // Preis des dritten Punktes in %
input color    InpColor=clrRed;        // Objektfarbe
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Linienstil
input int      InpWidth=2;             // Linienbreite
input bool     InpBack=false;          // Objekt im Hintergrund
input bool     InpSelection=true;       // Wählen um zu bewegen
input bool     InpRayLeft=false;       // Fortsetzung des Objekts nach links
input bool     InpRayRight=false;      // Fortsetzung des Objekts nach rechts
input bool     InpHidden=true;         // Ausgeblendet in der Objektliste
input long     InpZOrder=0;            // Priorität auf Mausklick

//+-----+
//| Erstellt Objekt Fibonacci Expansion auf angegebenen Koordinaten |
//+-----+

bool FiboExpansionCreate(const long      chart_ID=0,           // ID des Charts
                        const string    name="FiboExpansion", // Name
                        const int       sub_window=0,          // Nummer des Unt
                        datetime         time1=0,              // Zeit des erste
                        double           price1=0,              // Preis des erste
                        datetime         time2=0,              // Zeit des zweit
                        double           price2=0,              // Preis des zwei
                        datetime         time3=0,              // Zeit des dritt
                        double           price3=0,              // Preis des drit
                        const color      clr=clrRed,           // Objektfarbe
                        const ENUM_LINE_STYLE style=STYLE_SOLID, // Linienstil
                        const int        width=1,              // Linienbreite
                        const bool       back=false,           // Im Hintergrund
                        const bool       selection=true,        // Wählen um zu k
                        const bool       ray_left=false,        // Fortsetzung de
                        const bool       ray_right=false,       // Fortsetzung de
                        const bool       hidden=true,           // Ausgeblendet i
                        const long        z_order=0)            // Priorität auf

{
//--- Die Koordinaten der Bezugspunkte angeben, wenn sie nicht gesetzt sind
    ChangeFiboExpansionEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- "Fibonacci Expansion" auf angegebenen Koordinaten erstellen
    if(!ObjectCreate(chart_ID,name,OBJ_EXPANSION,sub_window,time1,price1,time2,price2,t
    {
        Print(__FUNCTION__,
              ": Objekt \" Fibonacci Expansion\" konnte nicht erstellt werden! Fehlercod
        return(false);
    }
};
//--- Objektfarbe setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- Linienstil setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- Linienbreite angeben
    ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);

```

```

//--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- aktivieren (true) oder deaktivieren (false) Bewegung des Objekts für Bewegung
//--- Wenn Sie ein grafisches Objekt durch Funktion ObjectCreate erstellen, das Objekt
//--- kann nicht ausgewählt und verschoben werden. Innerhalb dieser Methode ist der St
//--- true, was Sie das Objekt auswählen und verschieben erlaubt
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- aktivieren (true) oder deaktivieren (false) Fortsetzung des Objekts nach links
    ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- aktivieren (true) oder deaktivieren (false) Fortsetzung des Objekts nach rechts
    ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der
    ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- setzen die Priorität für eine Mausklick-Ereignisse auf dem Chart
    ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
    return(true);
; }
//+-----+
//| Gibt die Anzahl der Ebenen und ihren Parameter an |
//+-----+
bool FiboExpansionLevelsSet(int          levels,          // Anzahl der Ebenen
                           double       &values[],       // Werte der Linien
                           color        &colors[],       // Farbe der Linien
                           ENUM_LINE_STYLE &styles[],    // Linienstil
                           int          &widths[],       // Breite der Linien
                           const long    chart_ID=0,     // ID des Charts
                           const string  name="FiboExpansion") // Objektname
{
//--- Überprüfen wir die Größe des Arrays
if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
    levels!=ArraySize(widths) || levels!=ArraySize(values))
{
    Print(__FUNCTION__," : Länge des Arrays entspricht nicht der Anzahl der Ebenen, F
    return(false);
; }
//--- Die Anzahl der Ebenen angeben
    ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- die Eigenschaften der Ebenen im Zyklus angeben
    for(int i=0;i<levels;i++)
    {
        //--- Wert der Ebene
        ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
        //--- Farbe der Ebene
        ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
        //--- Stil der Ebene
        ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
        //--- Stärke der Ebene

```

```

        ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
        //--- Beschreibung der Ebene
        ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,"FE "+DoubleToString(100*value
        );
    };
//--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Verschiebt den Ankerpunkt von Fibonacci Expansion |
//+-----+
bool FiboExpansionPointChange(const long   chart_ID=0,           // ID des Charts
                             const string name="FiboExpansion", // Objektname
                             const int    point_index=0,        // Nummer des Punktes
                             datetime      time=0,              // Zeitkoordinate des
                             double         price=0)              // Preiskoordinate des

{
    //--- Wenn die Koordinaten des Punktes nicht angegeben sind, dann verschieben wir es a
    if(!time)
        time=TimeCurrent();
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- bewegen den Bezugspunkt
    if(!ObjectMove(chart_ID,name,point_index,time,price))
    {
        Print(__FUNCTION__,
              ": der Bezugspunkt konnte nicht bewegt werden! Fehlercode = ",GetLastError()
        );
        return(false);
    };
//--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Löscht das Objekt "Fibonacci Expansion" |
//+-----+
bool FiboExpansionDelete(const long   chart_ID=0,           // ID des Charts
                        const string name="FiboExpansion") // Objektname

{
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- das Objekt löschen
    if(!ObjectDelete(chart_ID,name))
    {
        Print(__FUNCTION__,
              ": Objekt \" Fibonacci Expansion\" konnte nicht gelöscht werden! Fehlercode = ",GetLastError()
        );
        return(false);
    };
//--- die erfolgreiche Umsetzung

```



```

    return(true);
; }

//+-----+
//| Überprüft die Werte der Ankerpunkten von Fibonacci Expansion und |
//| setzt Standardwerte für leere Werte                               |
//+-----+
void ChangeFiboExpansionEmptyPoints(datetime &time1,double &price1,datetime &time2,
                                     double &price2,datetime &time3,double &price3)
{
    //--- wenn die Zeit des dritten (rechten) Punktes nicht angegeben ist, wird sie auf de
    if(!time3)
        time3=TimeCurrent();
    //--- wenn der Preis des dritten Punktes nicht angegeben ist, wird er dem Bid-Wert gle
    if(!price3)
        price3=SymbolInfoDouble(Symbol(),SYMBOL_BID);
    //--- wenn die Zeit des ersten (linken) Punktes nicht angegeben ist, wird sie auf 9 Ba
    //--- Array um die Zeit der Öffnung der letzten 10 Bars zu empfangen
    datetime temp[];
    ArrayResize(temp,10);
    if(!time1)
    {
        CopyTime(Symbol(),Period(),time3,10,temp);
        //--- stellen wir den ersten Punkt 9 Baren links dem zweiten Punkt
        time1=temp[0];
    }
    //--- wenn der Preis des ersten Punktes nicht angegeben ist, wird er dem Preis des dri
    if(!price1)
        price1=price3;
    //--- wenn die Zeit des zweiten Punktes nicht angegeben ist, wird sie auf 7 Baren von
    if(!time2)
        time2=temp[2];
    //--- wenn der Preis des zweiten Punktes nicht angegeben ist, bewegen wir ihn 250 Pun
    if(!price2)
        price2=price1-250*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
; }

//+-----+
//| Script program start function                                     |
//+-----+
void OnStart()
{
    //--- Überprüfen die Richtigkeit der Eingabeparameter
    if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
        InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
        InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
    {
        Print("Fehler! Ungültige Eingabeparameter! ");
        return;
    }
; }

//--- Die Anzahl der sichtbaren Bars im Chart-Fenster

```

```

    int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- Größe des Arrays price
    int accuracy=1000;
//--- Arrays für Werte von Daten und Preise, die für Setzung und Änderung
//--- der Koordinaten der Ankerpunkten des Objekts verwendet werden
    datetime date[];
    double price[];
//--- Speicher reservieren
    ArrayResize(date,bars);
    ArrayResize(price,accuracy);
//--- Füllen die Anordnung von Daten
    ResetLastError();
    if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
    {
        Print("Kann nicht den Wert der Zeit kopieren! Fehlercode = ",GetLastError());
        return;
    }
//--- Füllen den Array der Preise
//--- den Minimal- und Maximalwert der Charts finden
    double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
    double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- Schritte der Preisänderung finden und das Array füllen
    double step=(max_price-min_price)/accuracy;
    for(int i=0;i<accuracy;i++)
        price[i]=min_price+i*step;
//--- Definieren die Punkte um Fibonacci Expansion zu zeichnen
    int d1=InpDate1*(bars-1)/100;
    int d2=InpDate2*(bars-1)/100;
    int d3=InpDate3*(bars-1)/100;
    int p1=InpPrice1*(accuracy-1)/100;
    int p2=InpPrice2*(accuracy-1)/100;
    int p3=InpPrice3*(accuracy-1)/100;
//--- Fibonacci Expansion erstellen
    if(!FiboExpansionCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],
        InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHider
    {
        return;
    }
//--- den Chart neu zeichnen und 1 Sekunde warten
    ChartRedraw();
    Sleep(1000);
//--- jetzt bewegen wir die Ankerpunkte
//--- Schleifenzähler
    int v_steps=accuracy/10;
//--- den ersten Punkt bewegen
    for(int i=0;i<v_steps;i++)
    {
        //--- den nächsten Wert nehmen
        if(p1>1)

```

```

        p1-=1;
        //--- den Punkt bewegen
        if(!FiboExpansionPointChange(0, InpName, 0, date[d1], price[p1]))
            return;
        //--- Überprüfen die Fakten von Zwangsabschaltung der Skript
        if(IsStopped())
            return;
        //--- den Chart neu zeichnen
        ChartRedraw();
    };
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//--- Schleifenzähler
    v_steps=accuracy/2;
//--- den dritten Punkt bewegen
    for(int i=0; i<v_steps; i++)
    {
        //--- den nächsten Wert nehmen
        if(p3>1)
            p3-=1;
        //--- den Punkt bewegen
        if(!FiboExpansionPointChange(0, InpName, 2, date[d3], price[p3]))
            return;
        //--- Überprüfen die Fakten von Zwangsabschaltung der Skript
        if(IsStopped())
            return;
        //--- den Chart neu zeichnen
        ChartRedraw();
    };
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//--- Schleifenzähler
    v_steps=accuracy*4/5;
//--- den zweiten Punkt bewegen
    for(int i=0; i<v_steps; i++)
    {
        //--- den nächsten Wert nehmen
        if(p2<accuracy-1)
            p2+=1;
        //--- den Punkt bewegen
        if(!FiboExpansionPointChange(0, InpName, 1, date[d2], price[p2]))
            return;
        //--- Überprüfen die Fakten von Zwangsabschaltung der Skript
        if(IsStopped())
            return;
        //--- den Chart neu zeichnen
        ChartRedraw();
    };
//--- 1 Sekunde Verzögerung

```

```
Sleep(1000);  
//--- das Objekt aus dem Chart löschen  
FiboExpansionDelete(0, InpName);  
ChartRedraw();  
//--- 1 Sekunde Verzögerung  
Sleep(1000);  
//---  
; }
```

## OBJ\_ELLIOTWAVE5

Elliott Impulswelle.



### Hinweis

Für "Elliott Impulswelle" können Sie Zeichnung der Linien zwischen den Punkten (Eigenschaft [OBJPROP\\_DRAWLINES](#)) aktivieren/deaktivieren, und auch die Höhe der Welle Kennzeichnung (Aufzählung [ENUM\\_ELLIOT\\_WAVE\\_DEGREE](#)) setzen.

### Beispiel

Das folgende Skript erstellt und bewegt auf dem Chart Objekt "Elliott Impulswelle". Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Das Skript erstellt ein graphisches Objekt \"Elliott Impulswelle\"
#property description "Koordinaten der Ankerpunkten werden in Pixeln angegeben"
#property description "Chartfenster."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="ElliottWave5";    // Objektname
input int         InpDate1=10;               // Datum des ersten Punktes in
input int         InpPrice1=90;              // Preis des ersten Punktes in
input int         InpDate2=20;               // Datum des zweiten Punktes in
input int         InpPrice2=40;              // Preis des zweiten Punktes in
input int         InpDate3=30;               // Datum des dritten Punktes in
```

```

input int          InpPrice3=60;           // Preis des dritten Punktes in
input int          InpDate4=40;           // Datum des vierten Punktes in
input int          InpPrice4=10;          // Preis des vierten Punktes in
input int          InpDate5=60;           // Datum des fünften Punktes in
input int          InpPrice5=40;          // Preis des fünften Punktes in
input ENUM_ELLIOT_WAVE_DEGREE InpDegree=ELLIOTT_MINOR; // Niveau
input bool         InpDrawLines=true;     // Anzeige der Linien
input color        InpColor=clrRed;       // Linienfarbe
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // Linienstil
input int          InpWidth=2;            // Linienbreite
input bool         InpBack=false;         // Objekt im Hintergrund
input bool         InpSelection=true;     // Wählen um zu bewegen
input bool         InpHidden=true;       // Ausgeblendet in der Objektlis
input long         InpZOrder=0;          // Priorität auf Mausklick
//+-----+
//| Erstellt Elliot Impulswelle auf angegebenen Koordinaten |
//+-----+

bool ElliotWave5Create(const long          chart_ID=0,           // ID de
                      const string        name="ElliotWave5",   // Name
                      const int           sub_window=0,         // Numme
                      datetime            time1=0,              // Zeit
                      double              price1=0,              // Preis
                      datetime            time2=0,              // Zeit
                      double              price2=0,              // Preis
                      datetime            time3=0,              // Zeit
                      double              price3=0,              // Preis
                      datetime            time4=0,              // Zeit
                      double              price4=0,              // Preis
                      datetime            time5=0,              // Zeit
                      double              price5=0,              // Preis
                      const ENUM_ELLIOT_WAVE_DEGREE degree=ELLIOTT_MINUETTE, // Grad
                      const bool          draw_lines=true,      // Anzei
                      const color         clr=clrRed,           // Objek
                      const ENUM_LINE_STYLE style=STYLE_SOLID,   // Linie
                      const int           width=1,              // Linie
                      const bool          back=false,           // Im Hi
                      const bool          selection=true,        // Wähle
                      const bool          hidden=true,           // Ausge
                      const long           z_order=0)            // Prior

{
//--- Die Koordinaten der Bezugspunkte angeben, wenn sie nicht gesetzt sind
    ChangeElliotWave5EmptyPoints(time1,price1,time2,price2,time3,price3,time4,price4,t
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- erstellt Elliott Impulswelle auf angegebenen Koordinaten
    if(!ObjectCreate(chart_ID,name,OBJ_ELLIOTWAVE5,sub_window,time1,price1,time2,price2,
        price3,time4,price4,time5,price5))
    {
        Print(__FUNCTION__,

```

```

        ": Objekt \"Elliott Impulswelle\" konnte nicht erstellt werden! Fehlercode
        return(false);
    };
//--- Grad (Wellengröße) setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_DEGREE,degree);
//--- aktivieren (true) oder deaktivieren (false) Anzeige der Linien
    ObjectSetInteger(chart_ID,name,OBJPROP_DRAWLINES,draw_lines);
//--- Objektfarbe setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- Linienstil setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- Linienbreite angeben
    ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- aktivieren (true) oder deaktivieren (false) Bewegung des Objekts für Bewegung
//--- Wenn Sie ein grafisches Objekt durch Funktion ObjectCreate erstellen, das Objekt
//--- kann nicht ausgewählt und verschoben werden. Innerhalb dieser Methode ist der St
//--- true, was Sie das Objekt auswählen und verschieben erlaubt
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der
    ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- setzen die Priorität für eine Mausklick-Ereignisse auf dem Chart
    ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Verschiebt den Ankerpunkt von "Elliott Impulswelle"          |
//+-----+
bool ElliotWave5PointChange(const long   chart_ID=0,          // ID des Charts
                           const string name="ElliotWave5",    // Objektname
                           const int    point_index=0,         // Nummer des Bezugspunkt
                           datetime     time=0,                // Zeitkoordinate des Bez
                           double        price=0)               // Preiskoordinate des Be
{
//--- Wenn die Koordinaten des Punktes nicht angegeben sind, dann verschieben wir es a
    if(!time)
        time=TimeCurrent();
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- bewegen den Bezugspunkt
    if(!ObjectMove(chart_ID,name,point_index,time,price))
    {
        Print(__FUNCTION__,
            ": der Bezugspunkt konnte nicht bewegt werden! Fehlercode = ",GetLastError

```

```

        return(false);
    };
}
//--- die erfolgreiche Umsetzung
return(true);
};
//+-----+
//| Löscht "Elliott Impulswelle" |
//+-----+
bool ElliotWave5Delete(const long   chart_ID=0,          // ID des Charts
                      const string name="ElliotWave5") // Objektname
{
//--- Setzen den Wert des Fehlers zurück
ResetLastError();
//--- das Objekt löschen
if(!ObjectDelete(chart_ID,name))
{
    Print(__FUNCTION__,
          ": Objekt \"Elliott Impulswelle\" konnte nicht gelöscht werden! Fehlercode
    return(false);
};
//--- die erfolgreiche Umsetzung
return(true);
};
//+-----+
//| Prüft die Werte der Ankerpunkten von "Elliott Impulswelle" |
//| und setzt Standardwerte für leere Werte |
//+-----+
void ChangeElliotWave5EmptyPoints(datetime &time1,double &price1,
                                   datetime &time2,double &price2,
                                   datetime &time3,double &price3,
                                   datetime &time4,double &price4,
                                   datetime &time5,double &price5)
{
//--- Array um die Zeit der Öffnung der letzten 10 Bars zu empfangen
datetime temp[];
ArrayResize(temp,10);
//--- Daten erhalten
CopyTime(Symbol(),Period(),TimeCurrent(),10,temp);
//--- den Wert von einem Pip auf dem aktuellen Chart erhalten
double point=SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- wenn die Zeit des ersten Punktes nicht angegeben ist, wird er auf 9 Balken auf c
if(!time1)
    time1=temp[0];
//--- Wenn der Preis des ersten Punktes nicht angegeben wird, dann wird es einen Wert
if(!price1)
    price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- wenn die Zeit des zweiten Punktes nicht angegeben ist, wird er auf 7 Balken auf
if(!time2)
    time2=temp[2];

```



```

//--- wenn der Preis des zweiten Punktes nicht angegeben ist, bewegen wir ihn 300 Punkt
    if(!price2)
        price2=price1-300*point;
//--- wenn die Zeit des dritten Punktes nicht angegeben ist, wird er auf 5 Balken auf
    if(!time3)
        time3=temp[4];
//--- wenn der Preis des dritten Punktes nicht angegeben ist, bewegen wir ihn 250 Punkt
    if(!price3)
        price3=price1-250*point;
//--- wenn die Zeit des vierten Punktes nicht angegeben ist, wird er auf 3 Balken auf
    if(!time4)
        time4=temp[6];
//--- wenn der Preis des vierten Punktes nicht angegeben ist, bewegen wir ihn 550 Punkt
    if(!price4)
        price4=price1-550*point;
//--- wenn die Zeit des fünften Punktes nicht angegeben ist, wird sie auf dem aktuellen
    if(!time5)
        time5=temp[9];
//--- wenn der Preis des fünften Punktes nicht angegeben ist, bewegen wir ihn 450 Punkt
    if(!price5)
        price5=price1-450*point;
; }
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//--- Überprüfen die Richtigkeit der Eingabeparameter
    if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
        InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
        InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100 ||
        InpDate4<0 || InpDate4>100 || InpPrice4<0 || InpPrice4>100 ||
        InpDate5<0 || InpDate5>100 || InpPrice5<0 || InpPrice5>100)
    {
        Print("Fehler! Ungültige Eingabeparameter! ");
        return;
    }
; }
//--- Die Anzahl der sichtbaren Bars im Chart-Fenster
    int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- Größe des Arrays price
    int accuracy=1000;
//--- Arrays für Werte von Daten und Preise, die für Setzung und Änderung
//--- der Koordinaten der Ankerpunkten des Objekts verwendet werden
    datetime date[];
    double price[];
//--- Speicher reservieren
    ArrayResize(date,bars);
    ArrayResize(price,accuracy);
//--- Füllen die Anordnung von Daten

```

```

ResetLastError();
if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
{
    Print("Kann nicht den Wert der Zeit kopieren! Fehlercode = ",GetLastError());
    return;
};
//--- Füllen den Array der Preise
//--- den Minimal- und Maximalwert der Charts finden
double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- Schritte der Preisänderung finden und das Array füllen
double step=(max_price-min_price)/accuracy;
for(int i=0;i<accuracy;i++)
    price[i]=min_price+i*step;
//--- Definieren die Punkte um "Elliott Impulswelle" zu zeichnen
int d1=InpDate1*(bars-1)/100;
int d2=InpDate2*(bars-1)/100;
int d3=InpDate3*(bars-1)/100;
int d4=InpDate4*(bars-1)/100;
int d5=InpDate5*(bars-1)/100;
int p1=InpPrice1*(accuracy-1)/100;
int p2=InpPrice2*(accuracy-1)/100;
int p3=InpPrice3*(accuracy-1)/100;
int p4=InpPrice4*(accuracy-1)/100;
int p5=InpPrice5*(accuracy-1)/100;
//--- Elliott Impulswelle erstellen
if(!ElliotWave5Create(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],p
    date[d4],price[p4],date[d5],price[p5],InpDegree,InpDrawLines,InpColor,InpStyle,1
    InpBack,InpSelection,InpHidden,InpZOrder))
{
    return;
};
//--- den Chart neu zeichnen und 1 Sekunde warten
ChartRedraw();
Sleep(1000);
//--- jetzt bewegen wir die Ankerpunkte
//--- Schleifenzähler
int v_steps=accuracy/5;
//--- den fünften Punkt bewegen
for(int i=0;i<v_steps;i++)
{
    //--- den nächsten Wert nehmen
    if(p5<accuracy-1)
        p5+=1;
    //--- den Punkt bewegen
    if(!ElliotWave5PointChange(0,InpName,4,date[d5],price[p5]))
        return;
}
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
if(IsStopped())

```

```

        return;
        //--- den Chart neu zeichnen
        ChartRedraw();
    };
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//--- Schleifenzähler
    v_steps=accuracy/5;
//--- den zweiten und dritten Punkt bewegen
    for(int i=0;i<v_steps;i++)
    {
        //--- die nächsten Werte nehmen
        if(p2<accuracy-1)
            p2+=1;
        if(p3>1)
            p3-=1;
        //--- Die Punkte bewegen
        if(!ElliotWave5PointChange(0,InpName,1,date[d2],price[p2]))
            return;
        if(!ElliotWave5PointChange(0,InpName,2,date[d3],price[p3]))
            return;
        //--- Überprüfen die Fakten von Zwangsabschaltung der Skript
        if(IsStopped())
            return;
        //--- den Chart neu zeichnen
        ChartRedraw();
    };
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//--- Schleifenzähler
    v_steps=accuracy*4/5;
//--- den ersten und vierten Punkt bewegen
    for(int i=0;i<v_steps;i++)
    {
        //--- die nächsten Werte nehmen
        if(p1>1)
            p1-=1;
        if(p4<accuracy-1)
            p4+=1;
        //--- Die Punkte bewegen
        if(!ElliotWave5PointChange(0,InpName,0,date[d1],price[p1]))
            return;
        if(!ElliotWave5PointChange(0,InpName,3,date[d4],price[p4]))
            return;
        //--- Überprüfen die Fakten von Zwangsabschaltung der Skript
        if(IsStopped())
            return;
        //--- den Chart neu zeichnen
        ChartRedraw();
    };

```

```
    ;}  
    //--- 1 Sekunde Verzögerung  
    Sleep(1000);  
    //--- das Objekt aus dem Chart löschen  
    ElliotWave5Delete(0, InpName);  
    ChartRedraw();  
    //--- 1 Sekunde Verzögerung  
    Sleep(1000);  
    //---  
    ;}
```

## OBJ\_ELLIOTWAVE3

Elliott Korrekturwelle.



### Hinweis

Für "Elliott Korrekturwelle" können Sie Zeichnung der Linien zwischen den Punkten (Eigenschaft [OBJPROP\\_DRAWLINES](#)) aktivieren/deaktivieren, und auch die Höhe der Welle Kennzeichnung (Aufzählung [ENUM\\_ELLIOT\\_WAVE\\_DEGREE](#)) setzen.

### Beispiel

Das folgende Skript erstellt und bewegt auf dem Chart Objekt "Elliott Korrekturwelle". Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Das Skript erstellt ein graphisches Objekt \"Elliott Korrekturwelle\"."
#property description "Koordinaten der Ankerpunkten werden als Prozentsatz der Größe des Charts angegeben."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="ElliottWave3";    // Objektname
input int         InpDate1=10;               // Datum des ersten Punktes in %
input int         InpPrice1=90;              // Preis des ersten Punktes in %
input int         InpDate2=30;               // Datum des zweiten Punktes in %
input int         InpPrice2=10;              // Preis des zweiten Punktes in %
input int         InpDate3=50;               // Datum des dritten Punktes in %
```

```

input int                InpPrice3=40;           // Preis des dritten Punktes in
input ENUM_ELLIOT_WAVE_DEGREE InpDegree=ELLIOTT_MINOR; // Niveau
input bool               InpDrawLines=true;      // Anzeige der Linien
input color              InpColor=clrRed;        // Linienfarbe
input ENUM_LINE_STYLE    InpStyle=STYLE_DASH;    // Linienstil
input int                InpWidth=2;            // Linienbreite
input bool               InpBack=false;          // Objekt im Hintergrund
input bool               InpSelection=true;      // Wählen um zu bewegen
input bool               InpHidden=true;         // Ausgeblendet in der Objektlis
input long               InpZOrder=0;           // Priorität auf Mausclick
//+-----+
//| Erstellt Elliot Korrekturwelle auf angegebenen Koordinaten |
//+-----+
bool ElliotWave3Create(const long      chart_ID=0,           // ID de
                      const string    name="ElliotWave3",   // Name
                      const int        sub_window=0,        // Numme
                      datetime         time1=0,             // Zeit
                      double           price1=0,            // Preis
                      datetime         time2=0,             // Zeit
                      double           price2=0,            // Preis
                      datetime         time3=0,             // Zeit
                      double           price3=0,            // Preis
                      const ENUM_ELLIOT_WAVE_DEGREE degree=ELLIOTT_MINUETTE, // Grad
                      const bool       draw_lines=true,     // Anzei
                      const color      clr=clrRed,         // Objek
                      const ENUM_LINE_STYLE style=STYLE_SOLID, // Linie
                      const int        width=1,            // Linie
                      const bool       back=false,         // Im H
                      const bool       selection=true,     // Wähle
                      const bool       hidden=true,        // Ausge
                      const long       z_order=0)          // Prior
{
//--- Die Koordinaten der Bezugspunkte angeben, wenn sie nicht gesetzt sind
    ChangeElliotWave3EmptyPoints(time1,price1,time2,price2,time3,price3);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- erstellt Elliott Korrekturwelle auf angegebenen Koordinaten
    if(!ObjectCreate(chart_ID,name,OBJ_ELLIOTWAVE3,sub_window,time1,price1,time2,price2,price3))
    {
        Print(__FUNCTION__,
              ": Objekt \"Elliott Korrekturwelle\" konnte nicht erstellt werden! Fehlercode: ",
              GetLastError(),
              "\n");
        return(false);
    }
//--- Grad (Wellengröße) setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_DEGREE,degree);
//--- aktivieren (true) oder deaktivieren (false) Anzeige der Linien
    ObjectSetInteger(chart_ID,name,OBJPROP_DRAWLINES,draw_lines);
//--- Objektfarbe setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);

```

```

//--- Linienstil setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- Linienbreite angeben
    ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- aktivieren (true) oder deaktivieren (false) Bewegung des Objekts für Bewegung
//--- Wenn Sie ein grafisches Objekt durch Funktion ObjectCreate erstellen, das Objekt
//--- kann nicht ausgewählt und verschoben werden. Innerhalb dieser Methode ist der St
//--- true, was Sie das Objekt auswählen und verschieben erlaubt
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der
    ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- setzen die Priorität für eine Mausklick-Ereignisse auf dem Chart
    ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
    return(true);
; }

//+-----+
//| Verschiebt den Ankerpunkt von "Elliott Korrekturwelle" |
//+-----+
bool ElliotWave3PointChange(const long   chart_ID=0,          // ID des Charts
                           const string name="ElliotWave3",    // Objektname
                           const int    point_index=0,         // Nummer des Bezugspunkt
                           datetime      time=0,               // Zeitkoordinate des Bez
                           double        price=0)               // Preiskoordinate des Be

{
//--- Wenn die Koordinaten des Punktes nicht angegeben sind, dann verschieben wir es a
    if(!time)
        time=TimeCurrent();
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- bewegen den Bezugspunkt
    if(!ObjectMove(chart_ID,name,point_index,time,price))
    {
        Print(__FUNCTION__,
              ": der Bezugspunkt konnte nicht bewegt werden! Fehlercode = ",GetLastError()
        );
        return(false);
    }
; }
//--- die erfolgreiche Umsetzung
    return(true);
; }

//+-----+
//| Löscht "Elliott Korrekturwelle" |
//+-----+
bool ElliotWave3Delete(const long   chart_ID=0,          // ID des Charts

```

```

        const string name="ElliotWave3") // Objektname
    {
//--- Setzen den Wert des Fehlers zurück
        ResetLastError();
//--- das Objekt löschen
        if(!ObjectDelete(chart_ID,name))
        {
            Print(__FUNCTION__,
                  ": Objekt \"Elliot Korrekturwelle\" konnte nicht gelöscht werden! Fehlercode: ",
                  GetLastError());
            return(false);
        }
//--- die erfolgreiche Umsetzung
        return(true);
    }
//+-----+
//| Prüft die Werte der Ankerpunkten von "Elliot Korrekturwelle" |
//| und setzt Standardwerte für leere Werte |
//+-----+
void ChangeElliotWave3EmptyPoints(datetime &time1,double &price1,
                                   datetime &time2,double &price2,
                                   datetime &time3,double &price3)
{
//--- Array um die Zeit der Öffnung der letzten 10 Bars zu empfangen
    datetime temp[];
    ArrayResize(temp,10);
//--- Daten erhalten
    CopyTime(Symbol(),Period(),TimeCurrent(),10,temp);
//--- den Wert von einem Pip auf dem aktuellen Chart erhalten
    double point=SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- wenn die Zeit des ersten Punktes nicht angegeben ist, wird er auf 9 Balken auf dem Chart gesetzt
    if(!time1)
        time1=temp[0];
//--- Wenn der Preis des ersten Punktes nicht angegeben wird, dann wird es einen Wert auf dem Chart gesetzt
    if(!price1)
        price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- wenn die Zeit des zweiten Punktes nicht angegeben ist, wird er auf 5 Balken auf dem Chart gesetzt
    if(!time2)
        time2=temp[4];
//--- wenn der Preis des zweiten Punktes nicht angegeben ist, bewegen wir ihn 300 Punkte vom ersten Punkt weg
    if(!price2)
        price2=price1-300*point;
//--- wenn die Zeit des dritten Punktes nicht angegeben ist, wird er auf 1 Balken auf dem Chart gesetzt
    if(!time3)
        time3=temp[8];
//--- wenn der Preis des dritten Punktes nicht angegeben ist, bewegen wir ihn 200 Punkte vom ersten Punkt weg
    if(!price3)
        price3=price1-200*point;
}
//+-----+

```



```

//| Script program start function |
//+-----+
void OnStart()
{
//--- Überprüfen die Richtigkeit der Eingabeparameter
    if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
        InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
        InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
    {
        Print("Fehler! Ungültige Eingabeparameter! ");
        return;
    };
//--- Die Anzahl der sichtbaren Bars im Chart-Fenster
    int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- Größe des Arrays price
    int accuracy=1000;
//--- Arrays für Werte von Daten und Preise, die für Setzung und Änderung
//--- der Koordinaten der Ankerpunkten des Objekts verwendet werden
    datetime date[];
    double price[];
//--- Speicher reservieren
    ArrayResize(date,bars);
    ArrayResize(price,accuracy);
//--- Füllen die Anordnung von Daten
    ResetLastError();
    if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
    {
        Print("Kann nicht den Wert der Zeit kopieren! Fehlercode = ",GetLastError());
        return;
    };
//--- Füllen den Array der Preise
//--- den Minimal- und Maximalwert der Charts finden
    double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
    double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- Schritte der Preisänderung finden und das Array füllen
    double step=(max_price-min_price)/accuracy;
    for(int i=0;i<accuracy;i++)
        price[i]=min_price+i*step;
//--- Definieren die Punkte um "Elliott Korrekturwelle" zu zeichnen
    int d1=InpDate1*(bars-1)/100;
    int d2=InpDate2*(bars-1)/100;
    int d3=InpDate3*(bars-1)/100;
    int p1=InpPrice1*(accuracy-1)/100;
    int p2=InpPrice2*(accuracy-1)/100;
    int p3=InpPrice3*(accuracy-1)/100;
//--- Elliott Korrekturwelle erstellen
    if(!ElliotWave3Create(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],p1,
        InpDegree,InpDrawLines,InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHider
    {

```

```

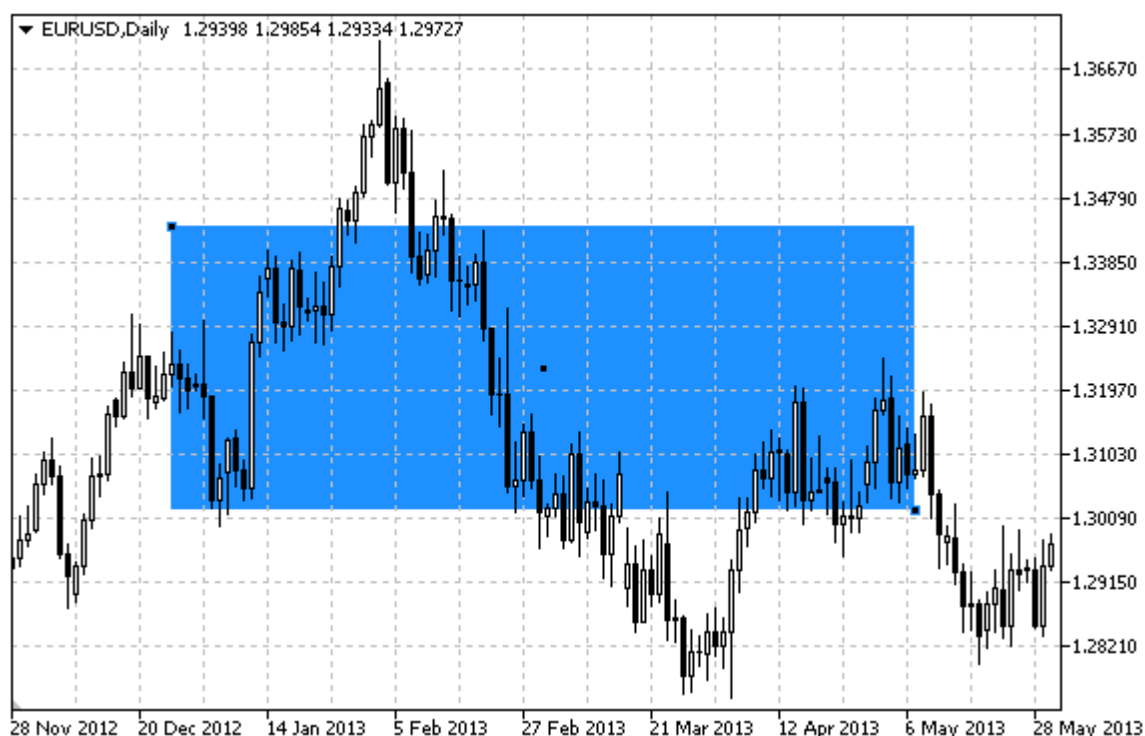
        return;
    };
//--- den Chart neu zeichnen und 1 Sekunde warten
    ChartRedraw();
    Sleep(1000);
//--- jetzt bewegen wir die Ankerpunkte
//--- Schleifenzähler
    int v_steps=accuracy/5;
//--- den dritten Punkt bewegen
    for(int i=0;i<v_steps;i++)
    {
        //--- den nächsten Wert nehmen
        if(p3<accuracy-1)
            p3+=1;
        //--- den Punkt bewegen
        if(!ElliotWave3PointChange(0,InpName,2,date[d3],price[p3]))
            return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
        if(IsStopped())
            return;
        //--- den Chart neu zeichnen
        ChartRedraw();
    };
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//--- Schleifenzähler
    v_steps=accuracy*4/5;
//--- den ersten und zweiten Punkt bewegen
    for(int i=0;i<v_steps;i++)
    {
        //--- die nächsten Werte nehmen
        if(p1>1)
            p1-=1;
        if(p2<accuracy-1)
            p2+=1;
        //--- Die Punkte bewegen
        if(!ElliotWave3PointChange(0,InpName,0,date[d1],price[p1]))
            return;
        if(!ElliotWave3PointChange(0,InpName,1,date[d2],price[p2]))
            return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
        if(IsStopped())
            return;
        //--- den Chart neu zeichnen
        ChartRedraw();
    };
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//--- das Objekt aus dem Chart löschen

```

```
ElliotWave3Delete(0, InpName);  
ChartRedraw();  
//--- 1 Sekunde Verzögerung  
Sleep(1000);  
//---  
; }
```

## OBJ\_RECTANGLE

Rechteck.



### Hinweis

Für das Rechteck, können Sie die Farbfüllung durch den Eigenschaften [OBJPROP\\_FILL](#) einstellen.

### Beispiel

Das folgende Skript erstellt und bewegt eine Rechteck auf dem Chart. Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Das Skript erstellt Rechteck auf dem Chart."
#property description "Koordinaten der Ankerpunkten werden als"
#property description "Prozentsatz der Größe des Chartfensters angegeben."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="Rectangle"; // Name des Rechtecks
input int         InpDate1=40;         // Datum des ersten Punktes in %
input int         InpPrice1=40;        // Preis des ersten Punktes in %
input int         InpDate2=60;         // Datum des zweiten Punktes in %
input int         InpPrice2=60;        // Preis des zweiten Punktes in %
input color       InpColor=clrRed;     // Farbe des Rechtecks
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // Linienstil des Rechtecks
input int         InpWidth=2;          // Linienbreite des Rechtecks
```

```

input bool      InpFill=true;      // Farbfüllung des Rechtecks
input bool      InpBack=false;     // Rechteck im Hintergrund
input bool      InpSelection=true; // Wählen zu bewegen
input bool      InpHidden=true;    // Ausgeblendet in der Objektliste
input long      InpZOrder=0;       // Priorität auf Mausclick
//+-----+
//| Erstellt ein Rechteck auf angegebenen Koordinaten |
//+-----+
bool RectangleCreate(const long      chart_ID=0,      // ID des Charts
                    const string     name="Rectangle", // Name des Rechtecks
                    const int        sub_window=0,    // Nummer des Unterfensters
                    datetime          time1=0,        // Zeit des ersten Punktes
                    double            price1=0,       // Preis des ersten Punktes
                    datetime          time2=0,        // Zeit des zweiten Punktes
                    double            price2=0,       // Preis des zweiten Punktes
                    const color       clr=clrRed,     // Farbe des Rechtecks
                    const ENUM_LINE_STYLE style=STYLE_SOLID, // Linienstil des Rechtecks
                    const int        width=1,        // Linienbreite des Rechtecks
                    const bool        fill=false,     // Farbfüllung des Rechtecks
                    const bool        back=false,     // In der Hintergrund
                    const bool        selection=true,  // Wählen um zu bewegen
                    const bool        hidden=true,    // Verborgen in der Liste der Objekte
                    const long        z_order=0)      // Priorität für Mausclick
{
//--- Die Koordinaten der Bezugspunkte angeben, wenn sie nicht gesetzt sind
    ChangeRectangleEmptyPoints(time1,price1,time2,price2);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Rechteck auf angegebenen Koordinaten erstellen
    if(!ObjectCreate(chart_ID,name,OBJ_RECTANGLE,sub_window,time1,price1,time2,price2))
    {
        Print(__FUNCTION__,
              ": konnte nicht Rechteck erstellen! Fehlercode = ",GetLastError());
        return(false);
    }
//--- Farbe des Rechtecks setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- Stil der Rechtecklinien einstellen
    ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- Linienbreite des Rechtecks setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- aktivieren (true) oder deaktivieren (false) Farbfüllung des Rechtecks
    ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- aktivieren (true) oder deaktivieren (false) Wählen des Rechtecks für Bewegung
//--- Wenn Sie ein grafisches Objekt durch Funktion ObjectCreate erstellen, das Objekt
//--- kann nicht ausgewählt und verschoben werden. Innerhalb dieser Methode ist der Stil
//--- true, was Sie das Objekt auswählen und verschieben erlaubt

```

```

    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der
    ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- setzen die Priorität für eine Mausklick-Ereignisse auf dem Chart
    ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Verschiebt den Ankerpunkt des Rechtecks |
//+-----+
bool RectanglePointChange(const long   chart_ID=0,      // ID des Charts
                          const string name="Rectangle", // Name des Rechtecks
                          const int    point_index=0,   // Nummer des Ankerpunktes
                          datetime      time=0,         // Zeitkoordinate des Punktes
                          double         price=0)        // Preiskoordinate des Punktes
{
//--- Wenn die Koordinaten des Punktes nicht angegeben sind, dann verschieben wir es auf
    if(!time)
        time=TimeCurrent();
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- bewegen den Bezugspunkt
    if(!ObjectMove(chart_ID,name,point_index,time,price))
    {
        Print(__FUNCTION__,
              ": der Bezugspunkt konnte nicht bewegt werden! Fehlercode = ",GetLastError());
        return(false);
    }
};
//--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Löscht das Rechteck |
//+-----+
bool RectangleDelete(const long   chart_ID=0,      // ID des Charts
                     const string name="Rectangle") // Name des Rechtecks
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- das Rechteck löschen
    if(!ObjectDelete(chart_ID,name))
    {
        Print(__FUNCTION__,
              ": konnte nicht Rechteck löschen! Fehlercode = ",GetLastError());
        return(false);
    }
};

```

```

    ;}
//--- die erfolgreiche Umsetzung
    return(true);
    ;}
//+-----+
//| Überprüft Werte der Ankerpunkten des Rechtecks |
//| und setzt Standardwerte für leere Werte          |
//+-----+
void ChangeRectangleEmptyPoints(datetime &time1,double &price1,
                                datetime &time2,double &price2)
{
//--- Wenn Zeit des ersten Punktes nicht angegeben ist, wird es auf dem aktuellen Bar
    if(!time1)
        time1=TimeCurrent();
//--- Wenn der Preis des ersten Punktes nicht angegeben wird, dann wird es einen Wert
    if(!price1)
        price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- wenn die Zeit des zweiten Punktes nicht angegeben ist, wird sie auf 9 Baren nach
    if(!time2)
    {
        //--- Array um Eröffnungszeit der letzten 10 Bars aufzunehmen
        datetime temp[10];
        CopyTime(Symbol(),Period(),time1,10,temp);
        //--- den zweiten Punkt 9 Bars nach links vom ersten Punkt setzen
        time2=temp[0];
    }
    ;}
//--- wenn der Preis des zweiten Punktes nicht angegeben ist, bewegen wir ihn 300 Punkt
    if(!price2)
        price2=price1-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
    ;}
//+-----+
//| Script program start function          |
//+-----+
void OnStart()
{
//--- Überprüfen die Richtigkeit der Eingabeparameter
    if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
        InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
    {
        Print("Fehler! Ungültige Eingabeparameter! ");
        return;
    }
    ;}
//--- Die Anzahl der sichtbaren Bars im Chart-Fenster
    int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- Größe des Arrays price
    int accuracy=1000;
//--- Arrays für Werte von Daten und Preise, die für Setzung und Änderung
//--- der Koordinaten der Ankerpunkten des Rechtecks verwendet werden
    datetime date[];

```

```

    double price[];
//--- Speicher reservieren
    ArrayResize(date,bars);
    ArrayResize(price,accuracy);
//--- Füllen die Anordnung von Daten
    ResetLastError();
    if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
    {
        Print("Kann nicht den Wert der Zeit kopieren! Fehlercode = ",GetLastError());
        return;
    }
//--- Füllen den Array der Preise
//--- den Minimal- und Maximalwert der Charts finden
    double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
    double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- Schritte der Preisänderung finden und das Array füllen
    double step=(max_price-min_price)/accuracy;
    for(int i=0;i<accuracy;i++)
        price[i]=min_price+i*step;
//--- Definieren die Punkte um ein Rechteck zu zeichnen
    int d1=InpDate1*(bars-1)/100;
    int d2=InpDate2*(bars-1)/100;
    int p1=InpPrice1*(accuracy-1)/100;
    int p2=InpPrice2*(accuracy-1)/100;
//--- Rechteck erstellen
    if(!RectangleCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor,
        InpStyle,InpWidth,InpFill,InpBack,InpSelection,InpHidden,InpZOrder))
    {
        return;
    }
//--- den Chart neu zeichnen und 1 Sekunde warten
    ChartRedraw();
    Sleep(1000);
//--- jetzt bewegen wir die Ankerpunkte des Rechtecks
//--- Schleifenzähler
    int h_steps=bars/2;
//--- die Punkte bewegen
    for(int i=0;i<h_steps;i++)
    {
        //--- die nächsten Werte nehmen
        if(d1<bars-1)
            d1+=1;
        if(d2>1)
            d2-=1;
        //--- Die Punkte bewegen
        if(!RectanglePointChange(0,InpName,0,date[d1],price[p1]))
            return;
        if(!RectanglePointChange(0,InpName,1,date[d2],price[p2]))
            return;
    }

```



```

//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
    if(IsStopped())
        return;
//--- den Chart neu zeichnen
    ChartRedraw();
// Verzögerung 0.05 Sekunden
    Sleep(50);
; }
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//--- Schleifenzähler
    int v_steps=accuracy/2;
//--- die Punkte bewegen
    for(int i=0;i<v_steps;i++)
    {
        //--- die nächsten Werte nehmen
        if(p1<accuracy-1)
            p1+=1;
        if(p2>1)
            p2-=1;
        //--- Die Punkte bewegen
        if(!RectanglePointChange(0, InpName, 0, date[d1], price[p1]))
            return;
        if(!RectanglePointChange(0, InpName, 1, date[d2], price[p2]))
            return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
        if(IsStopped())
            return;
        //--- den Chart neu zeichnen
        ChartRedraw();
    ; }
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//--- das Rechteck aus dem Chart löschen
    RectangleDelete(0, InpName);
    ChartRedraw();
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//---
; }

```

## OBJ\_TRIANGLE

Dreieck.



### Hinweis

Für das Dreieck, können Sie die Farbfüllung durch den Eigenschaften [OBJPROP\\_FILL](#) einstellen.

### Beispiel

Das folgende Skript erstellt und bewegt ein Dreieck auf dem Chart. Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Das Skript erstellt Dreieck auf dem Chart."
#property description "Koordinaten der Ankerpunkten werden als"
#property description "Prozentsatz der Größe des Chartfensters angegeben."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="Triangle";           // Name des Dreiecks
input int         InpDate1=25;                  // Datum des ersten Punktes in %
input int         InpPrice1=50;                 // Preis des ersten Punktes in %
input int         InpDate2=70;                 // Datum des zweiten Punktes in %
input int         InpPrice2=70;                // Preis des zweiten Punktes in %
input int         InpDate3=65;                 // Datum des dritten Punktes in %
input int         InpPrice3=20;                // Preis des dritten Punktes in %
input color       InpColor=clrRed;             // Farbe des Dreiecks
```

```

input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Linienstil des Dreiecks
input int              InpWidth=2;                // Linienbreite des Dreiecks
input bool             InpFill=false;             // Farbfüllung des Dreiecks
input bool             InpBack=false;             // Dreieck im Hintergrund
input bool             InpSelection=true;         // Wählen um zu bewegen
input bool             InpHidden=true;           // Ausgeblendet in der Objektliste
input long             InpZOrder=0;              // Priorität auf Mausclick
//+-----+
//| Erstellt ein Dreieck auf angegebenen Koordinaten |
//+-----+
bool TriangleCreate(const long      chart_ID=0,      // ID des Charts
                   const string    name="Triangle", // Name des Dreiecks
                   const int       sub_window=0,    // Nummer des Unterfensters
                   datetime         time1=0,        // Zeit des ersten Punktes
                   double           price1=0,       // Preis des ersten Punktes
                   datetime         time2=0,        // Zeit des zweiten Punktes
                   double           price2=0,       // Preis des zweiten Punktes
                   datetime         time3=0,        // Zeit des dritten Punktes
                   double           price3=0,       // Preis des dritten Punktes
                   const color      clr=clrRed,     // Farbe des Dreiecks
                   const ENUM_LINE_STYLE style=STYLE_SOLID, // Linienstil des Dreiecks
                   const int       width=1,        // Dreiecklinienbreite
                   const bool      fill=false,     // Farbfüllung des Dreiecks
                   const bool      back=false,    // In der Hintergrund
                   const bool      selection=true, // Wählen um zu bewegen
                   const bool      hidden=true,    // Verborgen in der Liste
                   const long      z_order=0)      // Priorität für Mausclick
{
    //--- Die Koordinaten der Bezugspunkte angeben, wenn sie nicht gesetzt sind
    ChangeTriangleEmptyPoints(time1,price1,time2,price2,time3,price3);
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Dreieck auf angegebenen Koordinaten erstellen
    if(!ObjectCreate(chart_ID,name,OBJ_TRIANGLE,sub_window,time1,price1,time2,price2,time3,price3))
    {
        Print(__FUNCTION__,
              ": konnte nicht Dreieck erstellen! Fehlercode = ",GetLastError());
        return(false);
    }
    //--- Farbe des Dreiecks setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
    //--- Stil der Dreiecklinien einstellen
    ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
    //--- Breite der Dreiecklinien einstellen
    ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
    //--- aktivieren (true) oder deaktivieren (false) Dreieckfüllung
    ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
    //--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
}

```

```

//--- aktivieren (true) oder deaktivieren (false) Wählen von Dreieck für Bewegung
//--- Wenn Sie ein grafisches Objekt durch Funktion ObjectCreate erstellen, das Objekt
//--- kann nicht ausgewählt und verschoben werden. Innerhalb dieser Methode ist der St
//--- true, was Sie das Objekt auswählen und verschieben erlaubt
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der
    ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- setzen die Priorität für eine Mausklick-Ereignisse auf dem Chart
    ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
    return(true);
; }

//+-----+
//| Verschiebt den Ankerpunkt des Dreiecks |
//+-----+
bool TrianglePointChange(const long   chart_ID=0,      // ID des Charts
                        const string name="Triangle",  // Name des Dreiecks
                        const int    point_index=0,    // Nummer des Ankerpunktes
                        datetime      time=0,          // Zeitkoordinate des Punktes
                        double        price=0)          // Preiskoordinate des Punktes
{
//--- Wenn die Koordinaten des Punktes nicht angegeben sind, dann verschieben wir es a
    if(!time)
        time=TimeCurrent();
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- bewegen den Bezugspunkt
    if(!ObjectMove(chart_ID,name,point_index,time,price))
    {
        Print(__FUNCTION__,
              ": der Bezugspunkt konnte nicht bewegt werden! Fehlercode = ",GetLastError());
        return(false);
    }
; }
//--- die erfolgreiche Umsetzung
    return(true);
; }

//+-----+
//| Löscht das Dreieck |
//+-----+
bool TriangleDelete(const long   chart_ID=0,      // ID des Charts
                   const string name="Triangle") // Name des Dreiecks
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Dreieck löschen
    if(!ObjectDelete(chart_ID,name))

```

```

    {
        Print(__FUNCTION__,
            ": Könnte nicht das Dreieck löschen! Fehlercode = ",GetLastError());
        return(false);
    }
}

//--- die erfolgreiche Umsetzung
return(true);
; }

//+-----+
//| Überprüft Werte der Ankerpunkten des Dreiecks |
//| und setzt Standardwerte für leere Werte |
//+-----+
void ChangeTriangleEmptyPoints(datetime &time1,double &price1,
                                datetime &time2,double &price2,
                                datetime &time3,double &price3)
{
    //--- Wenn Zeit des ersten Punktes nicht angegeben ist, wird es auf dem aktuellen Bar
    if(!time1)
        time1=TimeCurrent();
    //--- Wenn der Preis des ersten Punktes nicht angegeben wird, dann wird es einen Wert
    if(!price1)
        price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
    //--- wenn die Zeit des zweiten Punktes nicht angegeben ist, wird sie auf 9 Baren nach
    if(!time2)
    {
        //--- Array um Eröffnungszeit der letzten 10 Bars aufzunehmen
        datetime temp[10];
        CopyTime(Symbol(),Period(),time1,10,temp);
        //--- den zweiten Punkt 9 Bars nach links vom ersten Punkt setzen
        time2=temp[0];
    }
    ; }

    //--- wenn der Preis des zweiten Punktes nicht angegeben ist, bewegen wir ihn 300 Punkt
    if(!price2)
        price2=price1-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
    //--- wenn die Zeit des dritten Punktes nicht angegeben ist, wird sie der Zeit des zwe
    if(!time3)
        time3=time2;
    //--- wenn der Preis des ersten Punktes nicht angegeben ist, wird er dem Preis des ers
    if(!price3)
        price3=price1;
    ; }

//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    //--- Überprüfen die Richtigkeit der Eingabeparameter
    if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
        InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||

```

```

    InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)
    {
        Print("Fehler! Ungültige Eingabeparameter! ");
        return;
    }
}

//--- Die Anzahl der sichtbaren Bars im Chart-Fenster
int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- Größe des Arrays price
int accuracy=1000;
//--- Arrays für Werte von Daten und Preise, die für Setzung und Änderung
//--- der Koordinaten der Ankerpunkten des Dreiecks verwendet werden
datetime date[];
double price[];
//--- Speicher reservieren
ArrayResize(date,bars);
ArrayResize(price,accuracy);
//--- Füllen die Anordnung von Daten
ResetLastError();
if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
{
    Print("Kann nicht den Wert der Zeit kopieren! Fehlercode = ",GetLastError());
    return;
}
//--- Füllen den Array der Preise
//--- den Minimal- und Maximalwert der Charts finden
double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- Schritte der Preisänderung finden und das Array füllen
double step=(max_price-min_price)/accuracy;
for(int i=0;i<accuracy;i++)
    price[i]=min_price+i*step;
//--- Definieren die Punkte um ein Dreieck zu zeichnen
int d1=InpDate1*(bars-1)/100;
int d2=InpDate2*(bars-1)/100;
int d3=InpDate3*(bars-1)/100;
int p1=InpPrice1*(accuracy-1)/100;
int p2=InpPrice2*(accuracy-1)/100;
int p3=InpPrice3*(accuracy-1)/100;
//--- Dreieck erstellen
if(!TriangleCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],price[p3],
    InpColor,InpStyle,InpWidth,InpFill,InpBack,InpSelection,InpHidden,InpZOrder))
{
    return;
}
//--- den Chart neu zeichnen und 1 Sekunde warten
ChartRedraw();
Sleep(1000);
//--- jetzt bewegen wir die Ankerpunkte des Dreiecks
//--- Schleifenzähler

```

```

    int v_steps=accuracy*3/10;
    ///--- den ersten Punkt bewegen
    for(int i=0;i<v_steps;i++)
    {
        ///--- den nächsten Wert nehmen
        if(p1>1)
            p1-=1;
        ///--- den Punkt bewegen
        if(!TrianglePointChange(0,InpName,0,date[d1],price[p1]))
            return;
    }
    ///--- Überprüfen die Fakten von Zwangsabschaltung der Skript
    if(IsStopped())
        return;
    ///--- den Chart neu zeichnen
    ChartRedraw();
    ;}
    ///--- 1 Sekunde Verzögerung
    Sleep(1000);
    ///--- Schleifenzähler
    int h_steps=bars*9/20-1;
    ///--- den zweiten Punkt bewegen
    for(int i=0;i<h_steps;i++)
    {
        ///--- den nächsten Wert nehmen
        if(d2>1)
            d2-=1;
        ///--- den Punkt bewegen
        if(!TrianglePointChange(0,InpName,1,date[d2],price[p2]))
            return;
    }
    ///--- Überprüfen die Fakten von Zwangsabschaltung der Skript
    if(IsStopped())
        return;
    ///--- den Chart neu zeichnen
    ChartRedraw();
    // Verzögerung 0.05 Sekunden
    Sleep(50);
    ;}
    ///--- 1 Sekunde Verzögerung
    Sleep(1000);
    ///--- Schleifenzähler
    v_steps=accuracy/4;
    ///--- den dritten Punkt bewegen
    for(int i=0;i<v_steps;i++)
    {
        ///--- den nächsten Wert nehmen
        if(p3<accuracy-1)
            p3+=1;
        ///--- den Punkt bewegen
        if(!TrianglePointChange(0,InpName,2,date[d3],price[p3]))

```

```
        return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
    if(IsStopped())
        return;
//--- den Chart neu zeichnen
    ChartRedraw();
    ;}
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//--- das Dreieck aus dem Chart löschen
    TriangleDelete(0, InpName);
    ChartRedraw();
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//---
    ;}
```



## OBJ\_ELLIPSE

Ellipse.



### Hinweis

Für die Ellipse, können Sie die Farbfüllung durch den Eigenschaften [OBJPROP\\_FILL](#) einstellen.

### Beispiel

Das folgende Skript erstellt und bewegt eine Ellipse auf dem Chart. Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Das Skript erstellt Ellipse auf dem Chart."
#property description "Koordinaten der Ankerpunkten werden"
#property description "als Prozentsatz der Größe des Fensters angegeben"
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="Ellipse";           // Name der Ellipse
input int         InpDate1=30;                 // Datum des ersten Punktes in %
input int         InpPrice1=20;                // Preis des ersten Punktes in %
input int         InpDate2=70;                // Datum des zweiten Punktes in %
input int         InpPrice2=80;               // Preis des zweiten Punktes in %
input int         InpDate3=50;                // Datum des dritten Punktes in %
input int         InpPrice3=60;               // Preis des dritten Punktes in %
input color       InpColor=clrRed;             // Farbe der Ellipse
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Ellipselinienstil
```

```

input int      InpWidth=2;           // Linienbreite der Ellipse
input bool     InpFill=false;        // Farbfüllung der Ellipse
input bool     InpBack=false;        // Ellipse Im Hintergrund
input bool     InpSelection=true;    // Wählen um zu bewegen
input bool     InpHidden=true;       // Ausgeblendet in der Objektliste
input long     InpZOrder=0;          // Priorität auf Mausclick
//+-----+
//| Erstellt eine Ellipse auf angegebenen Koordinaten |
//+-----+
bool EllipseCreate(const long      chart_ID=0,      // ID des Charts
                   const string   name="Ellipse",  // Name der Ellipse
                   const int      sub_window=0,     // Nummer des Unterfensters
                   datetime       time1=0,          // Zeit des ersten Punktes
                   double          price1=0,        // Preis des ersten Punktes
                   datetime       time2=0,          // Zeit des zweiten Punktes
                   double          price2=0,        // Preis des zweiten Punktes
                   datetime       time3=0,          // Zeit des dritten Punktes
                   double          price3=0,        // Preis des dritten Punktes
                   const color     clr=clrRed,      // Farbe der Ellipse
                   const ENUM_LINE_STYLE style=STYLE_SOLID, // Ellipselinienstil
                   const int      width=1,         // Linienbreite der Ellipse
                   const bool     fill=false,      // Farbfüllung der Ellipse
                   const bool     back=false,      // Im Hintergrund
                   const bool     selection=true,   // Auswählen um zu bewegen
                   const bool     hidden=true,     // Ausgeblendet in der Obj
                   const long      z_order=0)       // Priorität auf Mausclick
{
    //--- Die Koordinaten der Bezugspunkte angeben, wenn sie nicht gesetzt sind
    ChangeEllipseEmptyPoints(time1,price1,time2,price2,time3,price3);
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Ellipse auf angegebenen Koordinaten erstellen
    if(!ObjectCreate(chart_ID,name,OBJ_ELLIPSE,sub_window,time1,price1,time2,price2,time3,price3))
    {
        Print(__FUNCTION__,
              ": konnte nicht Ellipse erstellen! Fehlercode = ",GetLastError());
        return(false);
    }
    ;}
    //--- Ellipsefarbe setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
    //--- Stil der Ellipselinien einstellen
    ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
    //--- Breite der Ellipselinien einstellen
    ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
    //--- aktivieren (true) oder deaktivieren (false) Ellipsefüllung
    ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
    //--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
    //--- aktivieren (true) oder deaktivieren (false) Wählen der Ellipse für Bewegung

```

```

//--- Wenn Sie ein grafisches Objekt durch Funktion ObjectCreate erstellen, das Objekt
//--- kann nicht ausgewählt und verschoben werden. Innerhalb dieser Methode ist der St
//--- true, was Sie das Objekt auswählen und verschieben erlaubt
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der
    ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- setzen die Priorität für eine Mausklick-Ereignisse auf dem Chart
    ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
    return(true);
; }

//+-----+
//| Verschiebt den Ankerpunkt der Ellipse |
//+-----+
bool EllipsePointChange(const long   chart_ID=0,      // ID des Charts
                        const string name="Ellipse",  // Name der Ellipse
                        const int    point_index=0,   // Nummer des Punktes
                        datetime      time=0,         // Zeitkoordinate des Punktes
                        double        price=0)         // Preiskoordinate des Punktes
{
//--- Wenn die Koordinaten des Punktes nicht angegeben sind, dann verschieben wir es a
    if(!time)
        time=TimeCurrent();
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- bewegen den Bezugspunkt
    if(!ObjectMove(chart_ID,name,point_index,time,price))
    {
        Print(__FUNCTION__,
              ": der Bezugspunkt konnte nicht bewegt werden! Fehlercode = ",GetLastError());
        return(false);
    }
; }
//--- die erfolgreiche Umsetzung
    return(true);
; }

//+-----+
//| Löscht die Ellipse |
//+-----+
bool EllipseDelete(const long   chart_ID=0,      // ID des Charts
                   const string name="Ellipse") // Name der Ellipse
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Ellipse löschen
    if(!ObjectDelete(chart_ID,name))
    {

```

```

        Print(__FUNCTION__,
              ": Könnte nicht die Ellipse löschen! Fehlercode = ", GetLastError());
        return(false);
    };
}

//--- die erfolgreiche Umsetzung
return(true);
};

//+-----+
//| Überprüft Werte der Ankerpunkten der Ellipse und für Leerwerte |
//| setzt Standardwerte |
//+-----+
void ChangeEllipseEmptyPoints(datetime &time1,double &price1,
                              datetime &time2,double &price2,
                              datetime &time3,double &price3)
{
    //--- Wenn Zeit des ersten Punktes nicht angegeben ist, wird es auf dem aktuellen Bar
    if(!time1)
        time1=TimeCurrent();
    //--- Wenn der Preis des ersten Punktes nicht angegeben wird, dann wird es einen Wert
    if(!price1)
        price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
    //--- wenn die Zeit des zweiten Punktes nicht angegeben ist, wird sie auf 9 Baren nach
    if(!time2)
    {
        //--- Array um Eröffnungszeit der letzten 10 Bars aufzunehmen
        datetime temp[10];
        CopyTime(Symbol(),Period(),time1,10,temp);
        //--- den zweiten Punkt 9 Bars nach links vom ersten Punkt setzen
        time2=temp[0];
    };
    //--- wenn der Preis des zweiten Punktes nicht angegeben ist, bewegen wir ihn 300 Punkt
    if(!price2)
        price2=price1-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
    //--- wenn die Zeit des dritten Punktes nicht angegeben ist, wird sie der Zeit des zwe
    if(!time3)
        time3=time2;
    //--- wenn der Preis des dritten Punktes nicht angegeben ist, wird er dem Preis des zw
    if(!price3)
        price3=price1;
};

//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    //--- Überprüfen die Richtigkeit der Eingabeparameter
    if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
        InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100 ||
        InpDate3<0 || InpDate3>100 || InpPrice3<0 || InpPrice3>100)

```

```

    {
        Print("Fehler! Ungültige Eingabeparameter! ");
        return;
    };
}

//--- Die Anzahl der sichtbaren Bars im Chart-Fenster
int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- Größe des Arrays price
int accuracy=1000;
//--- Arrays für Werte von Daten und Preise, die für Setzung und Änderung
//--- der Koordinaten der Ankerpunkten der Ellipse verwendet werden
datetime date[];
double price[];
//--- Speicher reservieren
ArrayResize(date,bars);
ArrayResize(price,accuracy);
//--- Füllen die Anordnung von Daten
ResetLastError();
if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
{
    Print("Kann nicht den Wert der Zeit kopieren! Fehlercode = ",GetLastError());
    return;
};
}

//--- Füllen den Array der Preise
//--- den Minimal- und Maximalwert der Charts finden
double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- Schritte der Preisänderung finden und das Array füllen
double step=(max_price-min_price)/accuracy;
for(int i=0;i<accuracy;i++)
    price[i]=min_price+i*step;
//--- Definieren die Punkte um die Ellipse zu zeichnen
int d1=InpDate1*(bars-1)/100;
int d2=InpDate2*(bars-1)/100;
int d3=InpDate3*(bars-1)/100;
int p1=InpPrice1*(accuracy-1)/100;
int p2=InpPrice2*(accuracy-1)/100;
int p3=InpPrice3*(accuracy-1)/100;
//--- Ellipse erstellen
if(!EllipseCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],price[
    InpColor,InpStyle,InpWidth,InpFill,InpBack,InpSelection,InpHidden,InpZOrder))
{
    return;
};
}

//--- den Chart neu zeichnen und 1 Sekunde warten
ChartRedraw();
Sleep(1000);
//--- jetzt bewegen wir die Ankerpunkte der Ellipse
//--- Schleifenzähler
int v_steps=accuracy/5;

```

```

//--- den ersten und zweiten Punkt bewegen
for(int i=0;i<v_steps;i++)
{
    //--- die nächsten Werte nehmen
    if(p1<accuracy-1)
        p1+=1;
    if(p2>1)
        p2-=1;
    //--- Die Punkte bewegen
    if(!EllipsePointChange(0,InpName,0,date[d1],price[p1]))
        return;
    if(!EllipsePointChange(0,InpName,1,date[d2],price[p2]))
        return;
    //--- Überprüfen die Fakten von Zwangsabschaltung der Skript
    if(IsStopped())
        return;
    //--- den Chart neu zeichnen
    ChartRedraw();
};
//--- 1 Sekunde Verzögerung
Sleep(1000);
//--- Schleifenzähler
int h_steps=bars/5;
//--- den dritten Punkt bewegen
for(int i=0;i<h_steps;i++)
{
    //--- den nächsten Wert nehmen
    if(d3>1)
        d3-=1;
    //--- den Punkt bewegen
    if(!EllipsePointChange(0,InpName,2,date[d3],price[p3]))
        return;
    //--- Überprüfen die Fakten von Zwangsabschaltung der Skript
    if(IsStopped())
        return;
    //--- den Chart neu zeichnen
    ChartRedraw();
    // Verzögerung 0.05 Sekunden
    Sleep(50);
};
//--- 1 Sekunde Verzögerung
Sleep(1000);
//--- die Ellipse aus dem Chart löschen
EllipseDelete(0,InpName);
ChartRedraw();
//--- 1 Sekunde Verzögerung
Sleep(1000);
//---
};

```

## OBJ\_ARROW\_THUMB\_UP

Zeichen "Gut".



### Hinweis

Die Position des Bezugspunktes relativ zum Zeichen kann aus der Aufzählung [ENUM\\_ARROW\\_ANCHOR](#) gewählt werden.

Größe Zeichen (mehr als 5) können nur durch Setzen der entsprechenden Eigenschaft [OBJPROP\\_WIDTH](#) beim Schreiben von Code in MetaEditor erstellt werden.

### Beispiel

Das folgende Skript erstellt und bewegt auf dem Chart Icons "Gut". Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Der Skript zieht Zeichen \"Gut\" (Daumen nach oben).\"
#property description "Koordinate des Bezugspunktes wird in Prozent von\"
#property description "der Größe des Chartfensters eingegeben.\"
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="ThumbUp";      // Name des Zeichens
input int         InpDate=75;              // Datum des Bezugspunktes in %
input int         InpPrice=25;             // Preis des Bezugspunktes in %
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_TOP; // Bezugsmethode
input color       InpColor=clrRed;        // Farbe des Zeichens
```

```

input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;    // Stil der Einfassungslinie
input int               InpWidth=5;            // Größe des Zeichens
input bool              InpBack=false;         // Das Zeichen im Hintergrund
input bool              InpSelection=true;      // Wählen zu bewegen
input bool              InpHidden=true;        // Ausblenden in der Liste der Objekten
input long              InpZOrder=0;           // Priorität auf Mausclick

//+-----+
//| Erstellt Zeichen "Gut"                                     |
//+-----+

bool ArrowThumbUpCreate(const long          chart_ID=0,          // ID des Charts
                        const string        name="ThumbUp",      // Name des Zeichens
                        const int           sub_window=0,        // Nummer des Unterfensters
                        datetime            time=0,              // Zeit des Bezugspunkts
                        double              price=0,              // Preis des Bezugspunkts
                        const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // Bezugsmethode
                        const color         clr=clrRed,          // Farbe des Zeichens
                        const ENUM_LINE_STYLE style=STYLE_SOLID, // Stil der Einfassungslinie
                        const int           width=3,             // Größe des Zeichens
                        const bool          back=false,          // Im Hintergrund anzeigen
                        const bool          selection=true,       // Wählen zu bewegen erlauben
                        const bool          hidden=true,          // Verborgen in der Liste der Objekte
                        const long          z_order=0)            // Priorität auf Mausclick
{
//--- Die Koordinaten des Bezugspunkts angeben, wenn sie nicht gesetzt sind
    ChangeArrowEmptyPoint(time,price);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Zeichen erstellen
    if(!ObjectCreate(chart_ID,name,OBJ_ARROW_THUMB_UP,sub_window,time,price))
    {
        Print(__FUNCTION__,
              ": Kann nicht Zeichen \"Gut\" erstellen! Fehlercode = ",GetLastError());
        return(false);
    }
//--- die Bindungsmethode setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- Die Farbe des Zeichens setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- Stil der Einfassungslinie setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- Die Größe des Zeichens setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- Aktivieren (true) oder deaktivieren (false) Mausbewegung Modus
//--- Wenn Sie ein grafisches Objekt durch Funktion ObjectCreate erstellen, das Objekt
//--- kann nicht ausgewählt und verschoben werden. Innerhalb dieser Methode ist der Stil
//--- true, was Sie das Objekt auswählen und verschieben erlaubt
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);

```



```

    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der
    ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- setzen die Priorität für eine Mausklick-Ereignisse auf dem Chart
    ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
    return(true);
; }

//+-----+
//| Bewegt den Bezugspunkt |
//+-----+
bool ArrowThumbUpMove(const long   chart_ID=0,      // ID des Charts
                     const string name="ThumbUp",  // Name des Objekts
                     datetime     time=0,          // Zeitkoordinate des Bezugspunkts
                     double        price=0)        // Preiskoordinate des Bezugspunkts
{
//--- Wenn die Koordinaten des Punktes nicht angegeben sind, dann verschieben wir es
    if(!time)
        time=TimeCurrent();
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- bewegen den Bezugspunkt
    if(!ObjectMove(chart_ID,name,0,time,price))
    {
        Print(__FUNCTION__,
              ": der Bezugspunkt konnte nicht bewegt werden! Fehlercode = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
; }

//+-----+
//| Ändert die Weise der Bindung von "Gut" |
//+-----+
bool ArrowThumbUpAnchorChange(const long   chart_ID=0,      // ID des Charts
                             const string name="ThumbUp",  // Objektname
                             const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // Bindungsme
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Bindungsmethode ändern
    if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
    {
        Print(__FUNCTION__,
              ": Bindungsmethode konnte nicht geändert werden! Fehlercode = ",GetLastError());
        return(false);
    }
; }

```

```

//--- die erfolgreiche Umsetzung
    return(true);
; }
//+-----+
//| Löscht das Zeichen "Gut" |
//+-----+
bool ArrowThumbUpDelete(const long   chart_ID=0,      // ID des Charts
                        const string name="ThumbUp") // Zeichenname
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Zeichen löschen
    if(!ObjectDelete(chart_ID,name))
    {
        Print(__FUNCTION__,
              ": Kann nicht Zeichen \"Gut\" löschen! Fehlercode = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
; }
//+-----+
//| Überprüft den Wert des Bezugspunkts für lehre Werte |
//| setzt Standardwerte |
//+-----+
void ChangeArrowEmptyPoint(datetime &time,double &price)
{
//--- Wenn Punktzeit nicht angegeben ist, wird es auf dem aktuellen Bar sein
    if(!time)
        time=TimeCurrent();
//--- Wenn der Preis des Punktes nicht angegeben wird, dann wird es einen Wert von Bid
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
; }
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//--- Überprüfen die Richtigkeit der Eingabeparameter
    if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
    {
        Print("Fehler! Ungültige Eingabeparameter! ");
        return;
    }
; }
//--- Die Anzahl der sichtbaren Bars im Chart-Fenster
    int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- Größe des Arrays price
    int accuracy=1000;

```

```

//--- Arrays für Werte von Daten und Preise, die für Setzung und Änderung
//--- der Koordinaten von Ankerpunkte verwendet werden
    datetime date[];
    double price[];
//--- Speicher reservieren
    ArrayResize(date,bars);
    ArrayResize(price,accuracy);
//--- Füllen die Anordnung von Daten
    ResetLastError();
    if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
    {
        Print("Kann nicht den Wert der Zeit kopieren! Fehlercode = ",GetLastError());
        return;
    }
//--- Füllen den Array der Preise
//--- den Minimal- und Maximalwert der Charts finden
    double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
    double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- Schritte der Preisänderung finden und das Array füllen
    double step=(max_price-min_price)/accuracy;
    for(int i=0;i<accuracy;i++)
        price[i]=min_price+i*step;
//--- Punkte für das Zeichnen der Zeichen definieren
    int d=InpDate*(bars-1)/100;
    int p=InpPrice*(accuracy-1)/100;
//--- erstellen das Zeichen "Gut" auf dem Chart
    if(!ArrowThumbUpCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
        InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
    {
        return;
    }
//--- den Chart neu zeichnen und 1 Sekunde warten
    ChartRedraw();
    Sleep(1000);
//--- Jetzt werden wir den Bezugspunkt bewegen und seine Position relativ zu dem Zeich
//--- Schleifenzähler
    int h_steps=bars/4;
//--- bewegen den Bezugspunkt
    for(int i=0;i<h_steps;i++)
    {
        //--- den nächsten Wert nehmen
        if(d>1)
            d-=1;
        //--- den Punkt bewegen
        if(!ArrowThumbUpMove(0,InpName,date[d],price[p]))
            return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
        if(IsStopped())
            return;
    }

```

```

    //--- den Chart neu zeichnen
    ChartRedraw();
    // Verzögerung 0.05 Sekunden
    Sleep(50);
    ;}
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//--- Schleifenzähler
    int v_steps=accuracy/4;
//--- bewegen den Bezugspunkt
    for(int i=0;i<v_steps;i++)
    {
        //--- den nächsten Wert nehmen
        if(p<accuracy-1)
            p+=1;
        //--- den Punkt bewegen
        if(!ArrowThumbUpMove(0, InpName, date[d], price[p]))
            return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
        if(IsStopped())
            return;
        //--- den Chart neu zeichnen
        ChartRedraw();
    }
//--- Die Position des Bezugspunktes relativ zum Zeichen ändern
    ArrowThumbUpAnchorChange(0, InpName, ANCHOR_BOTTOM);
//--- den Chart neu zeichnen
    ChartRedraw();
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//--- Zeichen aus dem Chart löschen
    ArrowThumbUpDelete(0, InpName);
    ChartRedraw();
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//---
    ;}

```

## OBJ\_ARROW\_THUMB\_DOWN

Zeichen "Schlecht".



### Hinweis

Die Position des Bezugspunktes relativ zum Zeichen kann aus der Aufzählung [ENUM\\_ARROW\\_ANCHOR](#) gewählt werden.

Größe Zeichen (mehr als 5) können nur durch Setzen der entsprechenden Eigenschaft [OBJPROP\\_WIDTH](#) beim Schreiben von Code in MetaEditor erstellt werden.

### Beispiel

Das folgende Skript erstellt und bewegt auf dem Chart Icons "Schlecht". Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Der Skript zieht Zeichen \"Schlecht\" (Daumen nach unten).\"
#property description "Koordinate des Bezugspunktes wird in Prozent von\"
#property description "der Größe des Chartfensters eingegeben.\"
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="ThumbDown";      // Name des Zeichens
input int         InpDate=25;                // Datum des Bezugspunktes in %
input int         InpPrice=75;               // Preis des Bezugspunktes in %
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_BOTTOM; // Bindungsmethode
```

```

input color          InpColor=clrRed;          // Farbe des Zeichens
input ENUM_LINE_STYLE InpStyle=STYLE_DOT;      // Stil der Einfassungslinie
input int            InpWidth=5;              // Größe des Zeichens
input bool           InpBack=false;           // Das Zeichen im Hintergrund
input bool           InpSelection=true;        // Wählen zu bewegen
input bool           InpHidden=true;          // Ausblenden in der Liste der Objekten
input long           InpZOrder=0;             // Priorität auf Mausclick

//+-----+
//| Erstellt Zeichen "Schlecht" |
//+-----+

bool ArrowThumbDownCreate(const long          chart_ID=0,          // ID des Char
                           const string      name="ThumbDown",    // Name des Ze
                           const int         sub_window=0,         // Nummer des Unter
                           datetime          time=0,               // Zeit des Bezugs
                           double            price=0,              // Preis des Bezug
                           const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // Bezugsmethode
                           const color       clr=clrRed,           // Farbe des Zeich
                           const ENUM_LINE_STYLE style=STYLE_SOLID, // Stil der Einfas
                           const int         width=3,              // Größe des Zeich
                           const bool        back=false,          // Im Hintergrund
                           const bool        selection=true,        // Wählen zu bewege
                           const bool        hidden=true,          // Verborgen in de
                           const long         z_order=0)            // Priorität auf M

{
//--- Die Koordinaten des Bezugspunkts angeben, wenn sie nicht gesetzt sind
    ChangeArrowEmptyPoint(time,price);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Zeichen erstellen
    if(!ObjectCreate(chart_ID,name,OBJ_ARROW_THUMB_DOWN,sub_window,time,price))
    {
        Print(__FUNCTION__,
              ": Kann nicht Zeichen \"Schlecht\" erstellen"! Fehlercode = ",GetLastError
        return(false);
    }
};
//--- die Bindungsmethode setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- Die Farbe des Zeichens setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- Stil der Einfassungslinie setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- Die Größe des Zeichens setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- Aktivieren (true) oder deaktivieren (false) Mausbewegung Modus
//--- Wenn Sie ein grafisches Objekt durch Funktion ObjectCreate erstellen, das Objekt
//--- kann nicht ausgewählt und verschoben werden. Innerhalb dieser Methode ist der St
//--- true, was Sie das Objekt auswählen und verschieben erlaubt

```

```

    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der
    ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- setzen die Priorität für eine Mausklick-Ereignisse auf dem Chart
    ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Bewegt den Bezugspunkt |
//+-----+
bool ArrowThumbDownMove(const long   chart_ID=0,      // ID des Charts
                        const string name="ThumbDown", // Name des Objekts
                        datetime      time=0,          // Zeitkoordinate des Bezugspunkts
                        double         price=0)         // Preiskoordinate des Bezugspunkts
{
//--- Wenn die Koordinaten des Punktes nicht angegeben sind, dann verschieben wir es
    if(!time)
        time=TimeCurrent();
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- bewegen den Bezugspunkt
    if(!ObjectMove(chart_ID,name,0,time,price))
    {
        Print(__FUNCTION__,
              ": der Bezugspunkt konnte nicht bewegt werden! Fehlercode = ",GetLastError());
        return(false);
    };
//--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Ändert die Weise der Bindung von "Schlecht" |
//+-----+
bool ArrowThumbDownAnchorChange(const long   chart_ID=0,      // ID des Charts
                                const string name="ThumbDown", // Name des Objekts
                                const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // Bindungsmethode
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Bindungsmethode ändern
    if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
    {
        Print(__FUNCTION__,
              ": Bindungsmethode konnte nicht geändert werden! Fehlercode = ",GetLastError());
        return(false);
    };
//--- die erfolgreiche Umsetzung
    return(true);
};

```

```

    ;}
//--- die erfolgreiche Umsetzung
    return(true);
    ;}
//+-----+
//| Löscht Zeichen "Schlecht" |
//+-----+
bool ArrowThumbDownDelete(const long   chart_ID=0,      // ID des Charts
                          const string name="ThumbDown") // Name des Zeichens
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Zeichen löschen
    if(!ObjectDelete(chart_ID,name))
    {
        Print(__FUNCTION__,
              ": Kann nicht Zeichen \"Schlecht\" löschen! Fehlercode = ",GetLastError());
        return(false);
    }
    ;}
//--- die erfolgreiche Umsetzung
    return(true);
    ;}
//+-----+
//| Überprüft den Wert des Bezugspunkts für lehre Werte |
//| setzt Standardwerte |
//+-----+
void ChangeArrowEmptyPoint(datetime &time,double &price)
{
//--- Wenn Punktzeit nicht angegeben ist, wird es auf dem aktuellen Bar sein
    if(!time)
        time=TimeCurrent();
//--- Wenn der Preis des Punktes nicht angegeben wird, dann wird es einen Wert von Bid
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
    ;}
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//--- Überprüfen die Richtigkeit der Eingabeparameter
    if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
    {
        Print("Fehler! Ungültige Eingabeparameter! ");
        return;
    }
    ;}
//--- Die Anzahl der sichtbaren Bars im Chart-Fenster
    int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- Größe des Arrays price

```



```

    int accuracy=1000;
    //--- Arrays für Werte von Daten und Preise, die für Setzung und Änderung
    //--- der Koordinaten von Ankerpunkte verwendet werden
    datetime date[];
    double price[];
    //--- Speicher reservieren
    ArrayResize(date,bars);
    ArrayResize(price,accuracy);
    //--- Füllen die Anordnung von Daten
    ResetLastError();
    if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
    {
        Print("Kann nicht den Wert der Zeit kopieren! Fehlercode = ",GetLastError());
        return;
    }
    //--- Füllen den Array der Preise
    //--- den Minimal- und Maximalwert der Charts finden
    double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
    double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
    //--- Schritte der Preisänderung finden und das Array füllen
    double step=(max_price-min_price)/accuracy;
    for(int i=0;i<accuracy;i++)
        price[i]=min_price+i*step;
    //--- Punkte für das Zeichnen der Zeichen definieren
    int d=InpDate*(bars-1)/100;
    int p=InpPrice*(accuracy-1)/100;
    //--- erstellen das Zeichen "Schlecht" auf dem Chart
    if(!ArrowThumbDownCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
        InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
    {
        return;
    }
    //--- den Chart neu zeichnen und 1 Sekunde warten
    ChartRedraw();
    Sleep(1000);
    //--- Jetzt werden wir den Bezugspunkt bewegen und seine Position relativ zu dem Zeich
    //--- Schleifenzähler
    int h_steps=bars/4;
    //--- bewegen den Bezugspunkt
    for(int i=0;i<h_steps;i++)
    {
        //--- den nächsten Wert nehmen
        if(d<bars-1)
            d+=1;
        //--- den Punkt bewegen
        if(!ArrowThumbDownMove(0,InpName,date[d],price[p]))
            return;
    }
    //--- Überprüfen die Fakten von Zwangsabschaltung der Skript
    if(IsStopped())

```

```

        return;
        //--- den Chart neu zeichnen
        ChartRedraw();
        // Verzögerung 0.05 Sekunden
        Sleep(50);
    };
    //--- 1 Sekunde Verzögerung
    Sleep(1000);
    //--- Schleifenzähler
    int v_steps=accuracy/4;
    //--- bewegen den Bezugspunkt
    for(int i=0;i<v_steps;i++)
    {
        //--- den nächsten Wert nehmen
        if(p>1)
            p-=1;
        //--- den Punkt bewegen
        if(!ArrowThumbDownMove(0,InpName,date[d],price[p]))
            return;
        //--- Überprüfen die Fakten von Zwangsabschaltung der Skript
        if(IsStopped())
            return;
        //--- den Chart neu zeichnen
        ChartRedraw();
    };
    //--- Die Position des Bezugspunktes relativ zum Zeichen ändern
    ArrowThumbDownAnchorChange(0,InpName,ANCHOR_TOP);
    //--- den Chart neu zeichnen
    ChartRedraw();
    //--- 1 Sekunde Verzögerung
    Sleep(1000);
    //--- Zeichen aus dem Chart löschen
    ArrowThumbDownDelete(0,InpName);
    ChartRedraw();
    //--- 1 Sekunde Verzögerung
    Sleep(1000);
    //---
    ;}

```

## OBJ\_ARROW\_UP

Zeichen "Pfeil nach oben".



### Hinweis

Die Position des Bezugspunktes relativ zum Zeichen kann aus der Aufzählung [ENUM\\_ARROW\\_ANCHOR](#) gewählt werden.

Größe Zeichen (mehr als 5) können nur durch Setzen der entsprechenden Eigenschaft [OBJPROP\\_WIDTH](#) beim Schreiben von Code in MetaEditor erstellt werden.

### Beispiel

Das folgende Skript erstellt und bewegt auf dem Chart Icons "Pfeil nach oben". Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Der Skript zieht Zeichen \"Pfeil nach oben\"."
#property description "Koordinate des Bezugspunktes wird als"
#property description "Prozentsatz der Größe des Chartfensters angegeben."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="ArrowUp";      // Name es Zeichens
input int         InpDate=25;              // Datum des Bezugspunktes in %
input int         InpPrice=25;             // Preis des Bezugspunktes in %
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_TOP; // Bezugsmethode
input color       InpColor=clrRed;         // Farbe des Zeichens
```

```

input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;    // Stil der Einfassungslinie
input int               InpWidth=5;            // Größe des Zeichens
input bool              InpBack=false;         // Das Zeichen im Hintergrund
input bool              InpSelection=false;     // Wählen zu bewegen
input bool              InpHidden=true;        // Ausblenden in der Liste der Objekten
input long              InpZOrder=0;           // Priorität auf Mausclick
//+-----+
//| Erstellt Zeichen "Pfeil nach oben" |
//+-----+
bool ArrowUpCreate(const long      chart_ID=0,      // ID des Charts
                  const string     name="ArrowUp",  // Name des Zeichens
                  const int        sub_window=0,    // Nummer des Unterfensters
                  datetime         time=0,          // Zeit des Bezugspunkts
                  double            price=0,         // Preis des Bezugspunkts
                  const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // Bezugsmethode
                  const color       clr=clrRed,     // Farbe des Zeichens
                  const ENUM_LINE_STYLE style=STYLE_SOLID, // Stil der Einfassungslinie
                  const int         width=3,        // Größe des Zeichens
                  const bool        back=false,     // Im Hintergrund
                  const bool        selection=true,  // Wählen zu bewegen
                  const bool        hidden=true,    // Verborgen in der Liste
                  const long         z_order=0)      // Priorität auf Mausclick
{
//--- Die Koordinaten des Bezugspunkts angeben, wenn sie nicht gesetzt sind
    ChangeArrowEmptyPoint(time,price);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Zeichen erstellen
    if(!ObjectCreate(chart_ID,name,OBJ_ARROW_UP,sub_window,time,price))
    {
        Print(__FUNCTION__,
              ": Kann nicht Zeichen \"Pfeil nach oben\" erstellen! Fehlercode = ",GetLastError());
        return(false);
    }
//--- die Bindungsmethode setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- Die Farbe des Zeichens setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- Stil der Einfassungslinie setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- Die Größe des Zeichens setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- Aktivieren (true) oder deaktivieren (false) Mausbewegung Modus
//--- Wenn Sie ein grafisches Objekt durch Funktion ObjectCreate erstellen, das Objekt
//--- kann nicht ausgewählt und verschoben werden. Innerhalb dieser Methode ist der Stil
//--- true, was Sie das Objekt auswählen und verschieben erlaubt
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);

```

```

    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der
    ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- setzen die Priorität für eine Mausklick-Ereignisse auf dem Chart
    ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
    return(true);
; }

//+-----+
//| Bewegt den Bezugspunkt |
//+-----+
bool ArrowUpMove(const long   chart_ID=0,      // ID des Charts
                 const string name="ArrowUp",  // Name des Objekts
                 datetime    time=0,          // Zeitkoordinate des Bezugspunkts
                 double       price=0)        // Preiskoordinate des Bezugspunkts
{
//--- Wenn die Koordinaten des Punktes nicht angegeben sind, dann verschieben wir es
    if(!time)
        time=TimeCurrent();
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- bewegen den Bezugspunkt
    if(!ObjectMove(chart_ID,name,0,time,price))
    {
        Print(__FUNCTION__,
              ": der Bezugspunkt konnte nicht bewegt werden! Fehlercode = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
; }

//+-----+
//| Ändert die Weise der Bindung von "Pfeil nach oben" |
//+-----+
bool ArrowUpAnchorChange(const long   chart_ID=0,      // ID des Charts
                        const string name="ArrowUp",  // Name des Objekts
                        const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // Bindungsmethode
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Ändern die Position des Bezugspunkts
    if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
    {
        Print(__FUNCTION__,
              ": Bindungsmethode konnte nicht geändert werden! Fehlercode = ",GetLastError());
        return(false);
    }
; }

```

```

//--- die erfolgreiche Umsetzung
    return(true);
; }
//+-----+
//| Löscht das Zeichen "Pfeil nach oben" |
//+-----+
bool ArrowUpDelete(const long chart_ID=0, // ID des Charts
                  const string name="ArrowUp") // Name des Zeichens
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Zeichen löschen
    if(!ObjectDelete(chart_ID,name))
    {
        Print(__FUNCTION__,
              ": Kann nicht Zeichen \"Pfeil nach oben\" löschen"! Fehlercode = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
; }
//+-----+
//| Überprüft den Wert des Bezugspunkts für lehere Werte |
//| setzt Standardwerte |
//+-----+
void ChangeArrowEmptyPoint(datetime &time,double &price)
{
//--- Wenn Punktzeit nicht angegeben ist, wird es auf dem aktuellen Bar sein
    if(!time)
        time=TimeCurrent();
//--- Wenn der Preis des Punktes nicht angegeben wird, dann wird es einen Wert von Bid
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
; }
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//--- Überprüfen die Richtigkeit der Eingabeparameter
    if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
    {
        Print("Fehler! Ungültige Eingabeparameter! ");
        return;
    }
; }
//--- Die Anzahl der sichtbaren Bars im Chart-Fenster
    int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- Größe des Arrays price
    int accuracy=1000;

```

```

//--- Arrays für Werte von Daten und Preise, die für Setzung und Änderung
//--- der Koordinaten von Ankerpunkte verwendet werden
    datetime date[];
    double price[];
//--- Speicher reservieren
    ArrayResize(date,bars);
    ArrayResize(price,accuracy);
//--- Füllen die Anordnung von Daten
    ResetLastError();
    if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
    {
        Print("Kann nicht den Wert der Zeit kopieren! Fehlercode = ",GetLastError());
        return;
    }
//--- Füllen den Array der Preise
//--- den Minimal- und Maximalwert der Charts finden
    double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
    double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- Schritte der Preisänderung finden und das Array füllen
    double step=(max_price-min_price)/accuracy;
    for(int i=0;i<accuracy;i++)
        price[i]=min_price+i*step;
//--- Punkte für das Zeichnen der Zeichen definieren
    int d=InpDate*(bars-1)/100;
    int p=InpPrice*(accuracy-1)/100;
//--- erstellen das Zeichen "Pfeil nach oben" auf dem Chart
    if(!ArrowUpCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
        InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
    {
        return;
    }
//--- den Chart neu zeichnen und 1 Sekunde warten
    ChartRedraw();
    Sleep(1000);
//--- Jetzt werden wir den Bezugspunkt bewegen und seine Position relativ zu dem Zeich
//--- Schleifenzähler
    int v_steps=accuracy/2;
//--- bewegen den Bezugspunkt
    for(int i=0;i<v_steps;i++)
    {
        //--- den nächsten Wert nehmen
        if(p<accuracy-1)
            p+=1;
        //--- den Punkt bewegen
        if(!ArrowUpMove(0,InpName,date[d],price[p]))
            return;
    }
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
    if(IsStopped())
        return;

```

```
    //--- den Chart neu zeichnen
    ChartRedraw();
    ;}
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//--- Die Position des Bezugspunktes relativ zum Zeichen ändern
    ArrowUpAnchorChange(0, InpName, ANCHOR_BOTTOM);
//--- den Chart neu zeichnen
    ChartRedraw();
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//--- Zeichen aus dem Chart löschen
    ArrowUpDelete(0, InpName);
    ChartRedraw();
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//---
    ;}
```



## OBJ\_ARROW\_DOWN

Zeichen "Pfeil nach unten".



### Hinweis

Die Position des Bezugspunktes relativ zum Zeichen kann aus der Aufzählung [ENUM\\_ARROW\\_ANCHOR](#) gewählt werden.

Größe Zeichen (mehr als 5) können nur durch Setzen der entsprechenden Eigenschaft [OBJPROP\\_WIDTH](#) beim Schreiben von Code in MetaEditor erstellt werden.

### Beispiel

Das folgende Skript erstellt und bewegt auf dem Chart Icons "Pfeil nach unten". Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Der Skript zieht Zeichen \"Pfeil nach unten\"."
#property description "Koordinate des Bezugspunktes wird als"
#property description "Prozentsatz der Größe des Chartfensters angegeben."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="ArrowDown";      // Zeichenname
input int         InpDate=75;                // Datum des Bezugspunktes in %
input int         InpPrice=75;               // Preis des Bezugspunktes in %
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_BOTTOM; // Bindungsmethode
input color       InpColor=clrRed;          // Farbe des Zeichens
input ENUM_LINE_STYLE InpStyle=STYLE_DOT;   // Stil der Einfassungslinie
```

```

input int      InpWidth=5;           // Größe des Zeichens
input bool     InpBack=false;        // Das Zeichen im Hintergrund
input bool     InpSelection=false;    // Wählen zu bewegen
input bool     InpHidden=true;       // Ausblenden in der Liste der Objekten
input long     InpZOrder=0;          // Priorität auf Mausclick

//+-----+
//| Erstellt Zeichen "Pfeil nach unten" |
//+-----+

bool ArrowDownCreate(const long      chart_ID=0,           // ID des Charts
                    const string     name="ArrowDown",     // Name des Zeichers
                    const int        sub_window=0,         // Nummer des Unter
                    datetime         time=0,               // Zeit des Bezugs
                    double            price=0,             // Preis des Bezugs
                    const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // Bezugsmethode
                    const color      clr=clrRed,          // Farbe des Zeich
                    const ENUM_LINE_STYLE style=STYLE_SOLID, // Stil der Einfas
                    const int         width=3,            // Größe des Zeich
                    const bool        back=false,         // Im Hintergrund
                    const bool        selection=true,      // Wählen zu bewege
                    const bool        hidden=true,        // Verborgen in de
                    const long        z_order=0)          // Priorität auf M

{
//--- Die Koordinaten des Bezugspunkts angeben, wenn sie nicht gesetzt sind
    ChangeArrowEmptyPoint(time,price);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Zeichen erstellen
    if(!ObjectCreate(chart_ID,name,OBJ_ARROW_DOWN,sub_window,time,price))
    {
        Print(__FUNCTION__,
              ": Kann nicht Zeichen \"Pfeil nach unten\" erstellen"! Fehlercode = ",GetLa
        return(false);
    };
//--- Bindungsmethode
    ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- Die Farbe des Zeichens setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- Stil der Einfassungslinie setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- Die Größe des Zeichens setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- Aktivieren (true) oder deaktivieren (false) Mausbewegung Modus
//--- Wenn Sie ein grafisches Objekt durch Funktion ObjectCreate erstellen, das Objekt
//--- kann nicht ausgewählt und verschoben werden. Innerhalb dieser Methode ist der St
//--- true, was Sie das Objekt auswählen und verschieben erlaubt
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);

```

```

//--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der
ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- setzen die Priorität für eine Mausklick-Ereignisse auf dem Chart
ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
return(true);
; }
//+-----+
//| Bewegt den Bezugspunkt |
//+-----+
bool ArrowDownMove(const long   chart_ID=0,      // ID des Charts
                  const string name="ArrowDown", // Name des Objekts
                  datetime      time=0,          // Zeitkoordinate des Bezugspunkts
                  double         price=0)        // Preiskoordinate des Bezugspunkts
{
//--- Wenn die Koordinaten des Punktes nicht angegeben sind, dann verschieben wir es
if(!time)
    time=TimeCurrent();
if(!price)
    price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- Setzen den Wert des Fehlers zurück
ResetLastError();
//--- bewegen den Bezugspunkt
if(!ObjectMove(chart_ID,name,0,time,price))
{
    Print(__FUNCTION__,
          ": der Bezugspunkt konnte nicht bewegt werden! Fehlercode = ",GetLastError());
    return(false);
; }
//--- die erfolgreiche Umsetzung
return(true);
; }
//+-----+
//| Ändert die Weise der Bindung von "Pfeil nach unten" |
//+-----+
bool ArrowDownAnchorChange(const long   chart_ID=0,      // ID des Charts
                          const string name="ArrowDown", // Name des Objekts
                          const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // Bindungsmethode
{
//--- Setzen den Wert des Fehlers zurück
ResetLastError();
//--- Ändern die Position des Bezugspunkts
if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
{
    Print(__FUNCTION__,
          ": Bindungsmethode konnte nicht geändert werden! Fehlercode = ",GetLastError());
    return(false);
; }
//--- die erfolgreiche Umsetzung

```

```

    return(true);
};

//+-----+
//| Löscht das Zeichen "Pfeil nach unten" |
//+-----+
bool ArrowDownDelete(const long   chart_ID=0,      // ID des Charts
                     const string name="ArrowDown") // Name des Zeichens
{
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Zeichen löschen
    if(!ObjectDelete(chart_ID,name))
    {
        Print(__FUNCTION__,
              ": Kann nicht Zeichen \"Pfeil nach unten\" löschen"! Fehlercode = ",GetLastError());
        return(false);
    }
};

//--- die erfolgreiche Umsetzung
return(true);
};

//+-----+
//| Überprüft den Wert des Bezugspunkts für leere Werte |
//| setzt Standardwerte |
//+-----+
void ChangeArrowEmptyPoint(datetime &time,double &price)
{
    //--- Wenn Punktzeit nicht angegeben ist, wird es auf dem aktuellen Bar sein
    if(!time)
        time=TimeCurrent();
    //--- Wenn der Preis des Punktes nicht angegeben wird, dann wird es einen Wert von Bid
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
};

//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    //--- Überprüfen die Richtigkeit der Eingabeparameter
    if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
    {
        Print("Fehler! Ungültige Eingabeparameter! ");
        return;
    }
};

//--- Die Anzahl der sichtbaren Bars im Chart-Fenster
int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- Größe des Arrays price
int accuracy=1000;
//--- Arrays für Werte von Daten und Preise, die für Setzung und Änderung

```

```

//--- der Koordinaten von Ankerpunkte verwendet werden
    datetime date[];
    double price[];
//--- Speicher reservieren
    ArrayResize(date,bars);
    ArrayResize(price,accuracy);
//--- Füllen die Anordnung von Daten
    ResetLastError();
    if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
    {
        Print("Kann nicht den Wert der Zeit kopieren! Fehlercode = ",GetLastError());
        return;
    };
//--- Füllen den Array der Preise
//--- den Minimal- und Maximalwert der Charts finden
    double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
    double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- Schritte der Preisänderung finden und das Array füllen
    double step=(max_price-min_price)/accuracy;
    for(int i=0;i<accuracy;i++)
        price[i]=min_price+i*step;
//--- Punkte für das Zeichnen der Zeichen definieren
    int d=InpDate*(bars-1)/100;
    int p=InpPrice*(accuracy-1)/100;
//--- erstellen das Zeichen "Pfeil nach unten" auf dem Chart
    if(!ArrowDownCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
        InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
    {
        return;
    };
//--- den Chart neu zeichnen und 1 Sekunde warten
    ChartRedraw();
    Sleep(1000);
//--- Jetzt werden wir den Bezugspunkt bewegen und seine Position relativ zu dem Zeich
//--- Schleifenzähler
    int v_steps=accuracy/2;
//--- bewegen den Bezugspunkt
    for(int i=0;i<v_steps;i++)
    {
        //--- den nächsten Wert nehmen
        if(p>1)
            p-=1;
        //--- den Punkt bewegen
        if(!ArrowDownMove(0,InpName,date[d],price[p]))
            return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
        if(IsStopped())
            return;
//--- den Chart neu zeichnen

```

```
        ChartRedraw();  
    };  
    //--- 1 Sekunde Verzögerung  
    Sleep(1000);  
    //--- Die Position des Bezugspunktes relativ zum Zeichen ändern  
    ArrowDownAnchorChange(0, InpName, ANCHOR_TOP);  
    //--- den Chart neu zeichnen  
    ChartRedraw();  
    //--- 1 Sekunde Verzögerung  
    Sleep(1000);  
    //--- Zeichen aus dem Chart löschen  
    ArrowDownDelete(0, InpName);  
    ChartRedraw();  
    //--- 1 Sekunde Verzögerung  
    Sleep(1000);  
    //---  
    ;}
```

## OBJ\_ARROW\_STOP

Zeichen "Stop".



### Hinweis

Die Position des Bezugspunktes relativ zum Zeichen kann aus der Aufzählung [ENUM\\_ARROW\\_ANCHOR](#) gewählt werden.

Größe Zeichen (mehr als 5) können nur durch Setzen der entsprechenden Eigenschaft [OBJPROP\\_WIDTH](#) beim Schreiben von Code in MetaEditor erstellt werden.

### Beispiel

Das folgende Skript erstellt und bewegt auf dem Chart Icons "Stop". Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Der Skript zieht Zeichen \"Stop\"."
#property description "Koordinate des Bezugspunktes wird als"
#property description "Prozentsatz der Größe des Chartfensters angegeben."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="ArrowStop";      // Name des Zeichens
input int         InpDate=10;               // Datum des Bezugspunktes in %
input int         InpPrice=50;              // Preis des Bezugspunktes in %
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_BOTTOM; // Bindungsmethode
input color       InpColor=clrRed;          // Farbe des Zeichens
```

```

input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;    // Stil der Einfassungslinie
input int                InpWidth=5;           // Größe des Zeichens
input bool               InpBack=false;        // Das Zeichen im Hintergrund
input bool               InpSelection=false;    // Wählen zu bewegen
input bool               InpHidden=true;       // Ausblenden in der Liste der Objekten
input long               InpZOrder=0;         // Priorität auf Mausklick

//+-----+
//| Erstellt das Zeichen "Stop"                                     |
//+-----+
bool ArrowStopCreate(const long          chart_ID=0,          // ID des Charts
                    const string         name="ArrowStop",    // Name des Zeichers
                    const int            sub_window=0,        // Nummer des Unter
                    datetime              time=0,             // Zeit des Bezugs
                    double                price=0,            // Preis des Bezug
                    const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // Bezugsmethode
                    const color           clr=clrRed,         // Farbe des Zeich
                    const ENUM_LINE_STYLE style=STYLE_SOLID,  // Stil der Einfas
                    const int              width=3,           // Größe des Zeich
                    const bool             back=false,        // Im Hintergrund
                    const bool             selection=true,     // Wählen zu bewege
                    const bool             hidden=true,       // Verborgen in de
                    const long             z_order=0)         // Priorität auf M

{
//--- Die Koordinaten des Bezugspunkts angeben, wenn sie nicht gesetzt sind
    ChangeArrowEmptyPoint(time,price);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Zeichen erstellen
    if(!ObjectCreate(chart_ID,name,OBJ_ARROW_STOP,sub_window,time,price))
    {
        Print(__FUNCTION__,
              ": Kann nicht Zeichen \"Stop\" erstellen! Fehlercode = ",GetLastError());
        return(false);
    }
//--- die Bindungsmethode setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- Die Farbe des Zeichens setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- Stil der Einfassungslinie setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- Die Größe des Zeichens setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- Aktivieren (true) oder deaktivieren (false) Mausbewegung Modus
//--- Wenn Sie ein grafisches Objekt durch Funktion ObjectCreate erstellen, das Objekt
//--- kann nicht ausgewählt und verschoben werden. Innerhalb dieser Methode ist der St
//--- true, was Sie das Objekt auswählen und verschieben erlaubt
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);

```



```

    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der
    ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- setzen die Priorität für eine Mausklick-Ereignisse auf dem Chart
    ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
    return(true);
; }

//+-----+
//| Bewegt den Bezugspunkt |
//+-----+
bool ArrowStopMove(const long   chart_ID=0,      // ID des Charts
                  const string name="ArrowStop", // Name des Objekts
                  datetime     time=0,          // Zeitkoordinate des Bezugspunkts
                  double        price=0)        // Preiskoordinate des Bezugspunkts
{
//--- Wenn die Koordinaten des Punktes nicht angegeben sind, dann verschieben wir es
    if(!time)
        time=TimeCurrent();
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- bewegen den Bezugspunkt
    if(!ObjectMove(chart_ID,name,0,time,price))
    {
        Print(__FUNCTION__,
              ": der Bezugspunkt konnte nicht bewegt werden! Fehlercode = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
; }

//+-----+
//| Ändert die Weise der Bindung von "Stop" |
//+-----+
bool ArrowStopAnchorChange(const long   chart_ID=0,      // ID des Charts
                          const string name="ArrowStop", // Name des Objekts
                          const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // Position des
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Bindungsmethode ändern
    if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
    {
        Print(__FUNCTION__,
              ": Bindungsmethode konnte nicht geändert werden! Fehlercode = ",GetLastError());
        return(false);
    }
; }

```

```

//--- die erfolgreiche Umsetzung
    return(true);
; }
//+-----+
//| Löscht das Zeichen "Stop" |
//+-----+
bool ArrowStopDelete(const long   chart_ID=0,      // ID des Charts
                    const string name="ArrowStop") // Name des Zeichens
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Zeichen löschen
    if(!ObjectDelete(chart_ID,name))
    {
        Print(__FUNCTION__,
              ": Kann nicht Zeichen \"Stop\" löschen"! Fehlercode = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
; }
//+-----+
//| Überprüft den Wert des Bezugspunkts für lehre Werte |
//| setzt Standardwerte |
//+-----+
void ChangeArrowEmptyPoint(datetime &time,double &price)
{
//--- Wenn Punktzeit nicht angegeben ist, wird es auf dem aktuellen Bar sein
    if(!time)
        time=TimeCurrent();
//--- Wenn der Preis des Punktes nicht angegeben wird, dann wird es einen Wert von Bid
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
; }
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//--- Überprüfen die Richtigkeit der Eingabeparameter
    if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
    {
        Print("Fehler! Ungültige Eingabeparameter! ");
        return;
    }
; }
//--- Die Anzahl der sichtbaren Bars im Chart-Fenster
    int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- Größe des Arrays price
    int accuracy=1000;

```

```

//--- Arrays für Werte von Daten und Preise, die für Setzung und Änderung
//--- der Koordinaten von Ankerpunkte verwendet werden
    datetime date[];
    double price[];
//--- Speicher reservieren
    ArrayResize(date,bars);
    ArrayResize(price,accuracy);
//--- Füllen die Anordnung von Daten
    ResetLastError();
    if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
    {
        Print("Kann nicht den Wert der Zeit kopieren! Fehlercode = ",GetLastError());
        return;
    }
//--- Füllen den Array der Preise
//--- den Minimal- und Maximalwert der Charts finden
    double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
    double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- Schritte der Preisänderung finden und das Array füllen
    double step=(max_price-min_price)/accuracy;
    for(int i=0;i<accuracy;i++)
        price[i]=min_price+i*step;
//--- Punkte für das Zeichnen der Zeichen definieren
    int d=InpDate*(bars-1)/100;
    int p=InpPrice*(accuracy-1)/100;
//--- erstellen das Zeichen "Stop" auf dem Chart
    if(!ArrowStopCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
        InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
    {
        return;
    }
//--- den Chart neu zeichnen und 1 Sekunde warten
    ChartRedraw();
    Sleep(1000);
//--- Jetzt werden wir den Bezugspunkt bewegen und seine Position relativ zu dem Zeich
//--- Schleifenzähler
    int h_steps=bars*2/5;
//--- bewegen den Bezugspunkt
    for(int i=0;i<h_steps;i++)
    {
        //--- den nächsten Wert nehmen
        if(d<bars-1)
            d+=1;
        //--- den Punkt bewegen
        if(!ArrowStopMove(0,InpName,date[d],price[p]))
            return;
    }
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
    if(IsStopped())
        return;

```

```

    //--- den Chart neu zeichnen
    ChartRedraw();
    // Verzögerung 0.025 Sekunden
    Sleep(25);
    ;}
//--- Die Position des Bezugspunktes relativ zum Zeichen ändern
    ArrowStopAnchorChange(0, InpName, ANCHOR_TOP);
//--- den Chart neu zeichnen
    ChartRedraw();
//--- Schleifenzähler
    h_steps=bars*2/5;
//--- bewegen den Bezugspunkt
    for(int i=0; i<h_steps; i++)
    {
        //--- den nächsten Wert nehmen
        if(d<bars-1)
            d+=1;
        //--- den Punkt bewegen
        if(!ArrowStopMove(0, InpName, date[d], price[p]))
            return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
        if(IsStopped())
            return;
        //--- den Chart neu zeichnen
        ChartRedraw();
        // Verzögerung 0.025 Sekunden
        Sleep(25);
    }
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//--- Zeichen aus dem Chart löschen
    ArrowStopDelete(0, InpName);
    ChartRedraw();
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//---
    ;}

```

## OBJ\_ARROW\_STOP

Zeichen "Checkmark".



### Hinweis

Die Position des Bezugspunktes relativ zum Zeichen kann aus der Aufzählung [ENUM\\_ARROW\\_ANCHOR](#) gewählt werden.

Größe Zeichen (mehr als 5) können nur durch Setzen der entsprechenden Eigenschaft [OBJPROP\\_WIDTH](#) beim Schreiben von Code in MetaEditor erstellt werden.

### Beispiel

Das folgende Skript erstellt und bewegt auf dem Chart Icons "Checkmark". Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Der Skript zieht Zeichen \"Checkmark\"."
#property description "Koordinate des Bezugspunktes wird als"
#property description "Prozentsatz der Größe des Chartfensters angegeben."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="ArrowCheck"; // Name des Zeichens
input int         InpDate=10;           // Datum des Bezugspunktes in %
input int         InpPrice=50;           // Preis des Bezugspunktes in %
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_TOP; // Bezugsmethode
input color       InpColor=clrRed;       // Farbe des Zeichens
```

```

input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;    // Stil der Einfassungslinie
input int                InpWidth=5;           // Größe des Zeichens
input bool              InpBack=false;         // Das Zeichen im Hintergrund
input bool              InpSelection=false;     // Wählen zu bewegen
input bool              InpHidden=true;        // Ausblenden in der Liste der Objekten
input long              InpZOrder=0;          // Priorität auf Mausklick
//+-----+
//| Erstellt Zeichen "Checkmark"                                     |
//+-----+
bool ArrowCheckCreate(const long          chart_ID=0,          // ID des Charts
                     const string        name="ArrowCheck",    // Name des Zeichens
                     const int           sub_window=0,         // Nummer des Unterfensters
                     datetime            time=0,              // Zeit des Bezugs
                     double              price=0,             // Preis des Bezugs
                     const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // Bezugsmethode
                     const color         clr=clrRed,          // Farbe des Zeichens
                     const ENUM_LINE_STYLE style=STYLE_SOLID,  // Stil der Einfassungslinie
                     const int           width=3,             // Größe des Zeichens
                     const bool          back=false,          // Im Hintergrund anzeigen
                     const bool          selection=true,       // Wählen zu bewegen
                     const bool          hidden=true,          // Verborgen in der Liste der Objekte
                     const long          z_order=0)           // Priorität auf Mausklick
{
//--- Die Koordinaten des Bezugspunkts angeben, wenn sie nicht gesetzt sind
    ChangeArrowEmptyPoint(time,price);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Zeichen erstellen
    if(!ObjectCreate(chart_ID,name,OBJ_ARROW_CHECK,sub_window,time,price))
    {
        Print(__FUNCTION__,
              ": Kann nicht Zeichen \"Checkmark\" erstellen"! Fehlercode = ",GetLastError());
        return(false);
    }
//--- die Bindungsmethode setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- Die Farbe des Zeichens setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- Stil der Einfassungslinie setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- Die Größe des Zeichens setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- Aktivieren (true) oder deaktivieren (false) Mausbewegung Modus
//--- Wenn Sie ein grafisches Objekt durch Funktion ObjectCreate erstellen, das Objekt
//--- kann nicht ausgewählt und verschoben werden. Innerhalb dieser Methode ist der Stil
//--- true, was Sie das Objekt auswählen und verschieben erlaubt
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);

```

```

    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der
    ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- setzen die Priorität für eine Mausklick-Ereignisse auf dem Chart
    ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
    return(true);
; }

//+-----+
//| Bewegt den Bezugspunkt |
//+-----+
bool ArrowCheckMove(const long   chart_ID=0,      // ID des Charts
                   const string name="ArrowCheck", // Name des Objekts
                   datetime     time=0,          // Zeitkoordinate des Bezugspunktes
                   double        price=0)         // Preiskoordinate des Bezugspunktes
{
//--- Wenn die Koordinaten des Punktes nicht angegeben sind, dann verschieben wir es auf
    if(!time)
        time=TimeCurrent();
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- bewegen den Bezugspunkt
    if(!ObjectMove(chart_ID,name,0,time,price))
    {
        Print(__FUNCTION__,
              ": der Bezugspunkt konnte nicht bewegt werden! Fehlercode = ",GetLastError());
        return(false);
    }
; }
//--- die erfolgreiche Umsetzung
    return(true);
; }

//+-----+
//| Ändert die Weise der Bindung von "Checkmark" |
//+-----+
bool ArrowCheckAnchorChange(const long   chart_ID=0,      // ID des Charts
                           const string name="ArrowCheck", // Name des Objekts
                           const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // Bindungsmethode
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Bindungsmethode ändern
    if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
    {
        Print(__FUNCTION__,
              ": Bindungsmethode konnte nicht geändert werden! Fehlercode = ",GetLastError());
        return(false);
    }
; }

```

```

//--- die erfolgreiche Umsetzung
    return(true);
; }
//+-----+
//| Löscht das Zeichen "Checkmark" |
//+-----+
bool ArrowCheckDelete(const long   chart_ID=0,          // ID des Charts
                      const string name="ArrowCheck") // Name des Zeichens
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Zeichen löschen
    if(!ObjectDelete(chart_ID,name))
    {
        Print(__FUNCTION__,
              ": Kann nicht Zeichen \"Checkmark\" löschen! Fehlercode = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
; }
//+-----+
//| Überprüft den Wert des Bezugspunkts für lehre Werte |
//| setzt Standardwerte |
//+-----+
void ChangeArrowEmptyPoint(datetime &time,double &price)
{
//--- Wenn Punktzeit nicht angegeben ist, wird es auf dem aktuellen Bar sein
    if(!time)
        time=TimeCurrent();
//--- Wenn der Preis des Punktes nicht angegeben wird, dann wird es einen Wert von Bid
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
; }
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//--- Überprüfen die Richtigkeit der Eingabeparameter
    if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
    {
        Print("Fehler! Ungültige Eingabeparameter! ");
        return;
    }
; }
//--- Die Anzahl der sichtbaren Bars im Chart-Fenster
    int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- Größe des Arrays price
    int accuracy=1000;

```



```

//--- Arrays für Werte von Daten und Preise, die für Setzung und Änderung
//--- der Koordinaten von Ankerpunkte verwendet werden
    datetime date[];
    double price[];
//--- Speicher reservieren
    ArrayResize(date,bars);
    ArrayResize(price,accuracy);
//--- Füllen die Anordnung von Daten
    ResetLastError();
    if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
    {
        Print("Kann nicht den Wert der Zeit kopieren! Fehlercode = ",GetLastError());
        return;
    }
//--- Füllen den Array der Preise
//--- den Minimal- und Maximalwert der Charts finden
    double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
    double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- Schritte der Preisänderung finden und das Array füllen
    double step=(max_price-min_price)/accuracy;
    for(int i=0;i<accuracy;i++)
        price[i]=min_price+i*step;
//--- Punkte für das Zeichnen der Zeichen definieren
    int d=InpDate*(bars-1)/100;
    int p=InpPrice*(accuracy-1)/100;
//--- erstellen das Zeichen "Checkmark" auf dem Chart
    if(!ArrowCheckCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
        InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
    {
        return;
    }
//--- den Chart neu zeichnen und 1 Sekunde warten
    ChartRedraw();
    Sleep(1000);
//--- Jetzt werden wir den Bezugspunkt bewegen und seine Position relativ zu dem Zeich
//--- Schleifenzähler
    int h_steps=bars*2/5;
//--- bewegen den Bezugspunkt
    for(int i=0;i<h_steps;i++)
    {
        //--- den nächsten Wert nehmen
        if(d<bars-1)
            d+=1;
        //--- den Punkt bewegen
        if(!ArrowCheckMove(0,InpName,date[d],price[p]))
            return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
        if(IsStopped())
            return;
    }

```

```

    //--- den Chart neu zeichnen
    ChartRedraw();
    // Verzögerung 0.025 Sekunden
    Sleep(25);
    ;}
//--- Die Position des Bezugspunktes relativ zum Zeichen ändern
    ArrowCheckAnchorChange(0, InpName, ANCHOR_BOTTOM);
//--- den Chart neu zeichnen
    ChartRedraw();
//--- Schleifenzähler
    h_steps=bars*2/5;
//--- bewegen den Bezugspunkt
    for(int i=0; i<h_steps; i++)
    {
        //--- den nächsten Wert nehmen
        if(d<bars-1)
            d+=1;
        //--- den Punkt bewegen
        if(!ArrowCheckMove(0, InpName, date[d], price[p]))
            return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
        if(IsStopped())
            return;
        //--- den Chart neu zeichnen
        ChartRedraw();
        // Verzögerung 0.025 Sekunden
        Sleep(25);
    }
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//--- Zeichen aus dem Chart löschen
    ArrowCheckDelete(0, InpName);
    ChartRedraw();
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//---
    ;}

```

## OBJ\_ARROW\_LEFT\_PRICE

Linkes Preisschild.



### Beispiel

Das folgende Skript erstellt und bewegt auf dem Chart ein linkes Preisschild. Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Der Skript erstellt ein linkes Preisschild auf dem Chart."
#property description "Koordinate des Bezugspunktes wird in Prozent"
#property description "von der Größe des Chartfensters eingegeben."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="LeftPrice";    // Name des Preisschilds
input int         InpDate=100;            // Datum des Bezugspunktes in %
input int         InpPrice=10;            // Preis des Bezugspunktes in %
input color       InpColor=clrRed;        // Farbe des Schilds
input ENUM_LINE_STYLE InpStyle=STYLE_SOLID; // Stil der Einfassungslinie
input int         InpWidth=2;             // Größe der Preisschilder
input bool        InpBack=false;          // Das Zeichen im Hintergrund
input bool        InpSelection=true;      // Wählen zu bewegen
input bool        InpHidden=true;        // Ausblenden in der Liste der Objekten
input long        InpZOrder=0;           // Priorität auf Mausclick
//+-----+
//| Erstellt ein linkes Preisschild |
```

```
//+-----+
bool ArrowLeftPriceCreate(const long      chart_ID=0,      // ID des Charts
                          const string    name="LeftPrice", // Name des Preisschildes
                          const int       sub_window=0,     // Nummer des Unterfensters
                          datetime         time=0,          // Zeit des Bezugspunktes
                          double           price=0,          // Preis des Bezugspunktes
                          const color      clr=clrRed,      // Farbe des Schildes
                          const ENUM_LINE_STYLE style=STYLE_SOLID, // Stil der Einfassung
                          const int       width=1,          // Größe des Schildes
                          const bool       back=false,      // In der Hintergrund
                          const bool       selection=true,   // Wählen zu bewegen
                          const bool       hidden=true,      // Verborgenen in der Liste
                          const long       z_order=0)        // Priorität für Mausclicks
{
    //--- Die Koordinaten des Bezugspunktes angeben, wenn sie nicht gesetzt sind
    ChangeArrowEmptyPoint(time,price);
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Ein Preisschild erstellen
    if(!ObjectCreate(chart_ID,name,OBJ_ARROW_LEFT_PRICE,sub_window,time,price))
    {
        Print(__FUNCTION__,
              ": Kann nicht das linke Preisschild erstellen! Fehlercode = ",GetLastError());
        return(false);
    }
    //--- Die Farbe des Schildes setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
    //--- Stil der Einfassungslinie setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
    //--- Die Größe des Schildes setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
    //--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
    //--- Aktivieren (true) oder deaktivieren (false) Mausbewegung Modus
    //--- Wenn Sie ein grafisches Objekt durch Funktion ObjectCreate erstellen, das Objekt
    //--- kann nicht ausgewählt und verschoben werden. Innerhalb dieser Methode ist der Stil
    //--- true, was Sie das Objekt auswählen und verschieben erlaubt
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
    //--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der Liste
    ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
    //--- setzen die Priorität für eine Mausclick-Ereignisse auf dem Chart
    ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
    //--- die erfolgreiche Umsetzung
    return(true);
}
//+-----+
//| Bewegt den Bezugspunkt |
//+-----+
```

```

bool ArrowLeftPriceMove(const long   chart_ID=0,          // ID des Charts
                        const string name="LeftPrice",    // Name des Schilds
                        datetime     time=0,              // Zeitkoordinate des Bezugspunkts
                        double        price=0)            // Preiskoordinate des Bezugspunkts
{
//--- Wenn die Koordinaten des Punktes nicht angegeben sind, dann verschieben wir es a
    if(!time)
        time=TimeCurrent();
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- bewegen den Bezugspunkt
    if(!ObjectMove(chart_ID,name,0,time,price))
    {
        Print(__FUNCTION__,
              ": der Bezugspunkt konnte nicht bewegt werden! Fehlercode = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
}

//+-----+
//| Löscht das linke Preisschild aus dem Chart |
//+-----+
bool ArrowLeftPriceDelete(const long   chart_ID=0,          // ID des Charts
                          const string name="LeftPrice")    // Name des Schilds
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- das Schild löschen
    if(!ObjectDelete(chart_ID,name))
    {
        Print(__FUNCTION__,
              ": Kann nicht das linke Preisschild löschen! Fehlercode = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
}

//+-----+
//| Überprüft den Wert des Bezugspunkts für lehre Werte |
//| setzt Standardwerte |
//+-----+
void ChangeArrowEmptyPoint(datetime &time,double &price)
{
//--- Wenn Punktzeit nicht angegeben ist, wird es auf dem aktuellen Bar sein
    if(!time)
        time=TimeCurrent();

```

```

//--- Wenn der Preis des Punktes nicht angegeben wird, dann wird es einen Wert von Bid
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
    ;}
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//--- Überprüfen die Richtigkeit der Eingabeparameter
    if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
    {
        Print("Fehler! Ungültige Eingabeparameter! ");
        return;
    }
//--- Die Anzahl der sichtbaren Bars im Chart-Fenster
    int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- Größe des Arrays price
    int accuracy=1000;
//--- Arrays für Werte von Daten und Preise, die für Setzung und Änderung
//--- der Koordinaten von Bezugspunkte des Preisschilds verwendet werden
    datetime date[];
    double price[];
//--- Speicher reservieren
    ArrayResize(date,bars);
    ArrayResize(price,accuracy);
//--- Füllen die Anordnung von Daten
    ResetLastError();
    if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
    {
        Print("Kann nicht den Wert der Zeit kopieren! Fehlercode = ",GetLastError());
        return;
    }
//--- Füllen den Array der Preise
//--- den Minimal- und Maximalwert der Charts finden
    double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
    double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- Schritte der Preisänderung finden und das Array füllen
    double step=(max_price-min_price)/accuracy;
    for(int i=0;i<accuracy;i++)
        price[i]=min_price+i*step;
//--- Punkte für das Zeichnen des Schilds definieren
    int d=InpDate*(bars-1)/100;
    int p=InpPrice*(accuracy-1)/100;
//--- erstellen das linke Preisschild auf dem Chart
    if(!ArrowLeftPriceCreate(0,InpName,0,date[d],price[p],InpColor,
        InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
    {
        return;
    }
}

```

```
    ;}
//--- den Chart neu zeichnen und 1 Sekunde warten
    ChartRedraw();
    Sleep(1000);
//--- Jetzt werden wir den Bezugspunkt bewegen
//--- Schleifenzähler
    int v_steps=accuracy*4/5;
//--- bewegen den Bezugspunkt
    for(int i=0;i<v_steps;i++)
    {
        //--- den nächsten Wert nehmen
        if(p<accuracy-1)
            p+=1;
        //--- den Punkt bewegen
        if(!ArrowLeftPriceMove(0,InpName,date[d],price[p]))
            return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
        if(IsStopped())
            return;
        //--- den Chart neu zeichnen
        ChartRedraw();
    }
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//--- Schild aus dem Chart löschen
    ArrowLeftPriceDelete(0,InpName);
    ChartRedraw();
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//---
    ;}
```

## OBJ\_ARROW\_RIGHT\_PRICE

Rechtes Preisschild.



### Beispiel

Das folgende Skript erstellt und bewegt auf dem Chart ein rechtes Preisschild. Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Der Skript erstellt ein rechtes Preisschild auf dem Chart."
#property description "Koordinate des Bezugspunktes wird in Prozent"
#property description "von der Größe des Chartfensters eingegeben."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="RightPrice"; // Name des Preisschilds
input int         InpDate=0;           // Datum des Bezugspunktes in %
input int         InpPrice=90;         // Preis des Bezugspunktes in %
input color       InpColor=clrRed;     // Farbe des Schilds
input ENUM_LINE_STYLE InpStyle=STYLE_SOLID; // Stil der Einfassungslinie
input int         InpWidth=2;          // Größe der Preisschilder
input bool        InpBack=false;       // Das Zeichen im Hintergrund
input bool        InpSelection=true;   // Wählen zu bewegen
input bool        InpHidden=true;      // Ausblenden in der Liste der Objekten
input long        InpZOrder=0;         // Priorität auf Mausclick
//+-----+
//| Erstellt ein rechtes Preisschild |
```



```
//+-----+
bool ArrowRightPriceCreate(const long      chart_ID=0,      // ID des Charts
                           const string    name="RightPrice", // Name des Preisschildes
                           const int      sub_window=0,    // Nummer des Unterfensters
                           datetime        time=0,          // Zeit des Bezugspunktes
                           double          price=0,          // Preis des Bezugspunktes
                           const color     clr=clrRed,      // Farbe des Schildes
                           const ENUM_LINE_STYLE style=STYLE_SOLID, // Stil der Einfassung
                           const int      width=1,          // Größe des Preisschildes
                           const bool      back=false,      // In der Hintergrund
                           const bool      selection=true,   // Wählen zu bewegen
                           const bool      hidden=true,     // Verborgenen in der Liste
                           const long      z_order=0)        // Priorität für Mausclicks
{
    //--- Die Koordinaten des Bezugspunktes angeben, wenn sie nicht gesetzt sind
    ChangeArrowEmptyPoint(time,price);
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Ein Preisschild erstellen
    if(!ObjectCreate(chart_ID,name,OBJ_ARROW_RIGHT_PRICE,sub_window,time,price))
    {
        Print(__FUNCTION__,
              ": Kann nicht das rechte Preisschild erstellen! Fehlercode = ",GetLastError());
        return(false);
    }
    ;}

    //--- Die Farbe des Schildes setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
    //--- Stil der Einfassungslinie setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
    //--- Die Größe des Schildes setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
    //--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
    //--- Aktivieren (true) oder deaktivieren (false) Mausbewegung Modus
    //--- Wenn Sie ein grafisches Objekt durch Funktion ObjectCreate erstellen, das Objekt
    //--- kann nicht ausgewählt und verschoben werden. Innerhalb dieser Methode ist der Stil
    //--- true, was Sie das Objekt auswählen und verschieben erlaubt
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
    //--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der Liste
    ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
    //--- setzen die Priorität für eine Mausclick-Ereignisse auf dem Chart
    ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
    //--- die erfolgreiche Umsetzung
    return(true);
    ;}
//+-----+
//| Bewegt den Bezugspunkt |
//+-----+
```

```

bool ArrowRightPriceMove(const long   chart_ID=0,           // ID des Charts
                        const string name="RightPrice", // Name des Schilds
                        datetime    time=0,           // Zeitkoordinate des Bezugspunktes
                        double       price=0)         // Preiskoordinate des Bezugspunktes
{
//--- Wenn die Koordinaten des Punktes nicht angegeben sind, dann verschieben wir es auf den aktuellen Bar
if(!time)
    time=TimeCurrent();
if(!price)
    price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- Setzen den Wert des Fehlers zurück
ResetLastError();
//--- bewegen den Bezugspunkt
if(!ObjectMove(chart_ID,name,0,time,price))
{
    Print(__FUNCTION__,
          ": der Bezugspunkt konnte nicht bewegt werden! Fehlercode = ",GetLastError());
    return(false);
};
//--- die erfolgreiche Umsetzung
return(true);
};
//+-----+
//| Löscht das rechte Preisschild aus dem Chart |
//+-----+
bool ArrowRightPriceDelete(const long   chart_ID=0,           // ID des Charts
                          const string name="RightPrice") // Name des Schilds
{
//--- Setzen den Wert des Fehlers zurück
ResetLastError();
//--- das Schild löschen
if(!ObjectDelete(chart_ID,name))
{
    Print(__FUNCTION__,
          ": Kann nicht das rechte Preisschild löschen! Fehlercode = ",GetLastError());
    return(false);
};
//--- die erfolgreiche Umsetzung
return(true);
};
//+-----+
//| Überprüft den Wert des Bezugspunktes für leere Werte |
//| setzt Standardwerte |
//+-----+
void ChangeArrowEmptyPoint(datetime &time,double &price)
{
//--- Wenn Punktzeit nicht angegeben ist, wird es auf dem aktuellen Bar sein
if(!time)
    time=TimeCurrent();

```

```

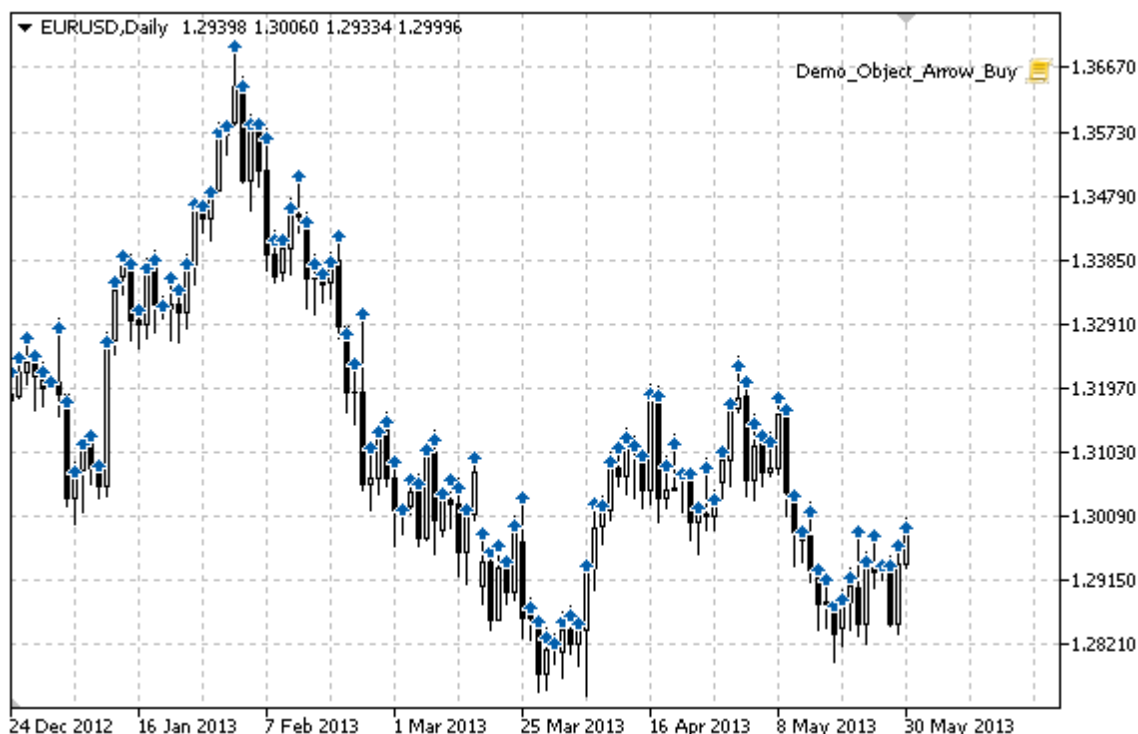
//--- Wenn der Preis des Punktes nicht angegeben wird, dann wird es einen Wert von Bid
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
; }
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//--- Überprüfen die Richtigkeit der Eingabeparameter
    if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
    {
        Print("Fehler! Ungültige Eingabeparameter! ");
        return;
    }
; }
//--- Die Anzahl der sichtbaren Bars im Chart-Fenster
    int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- Größe des Arrays price
    int accuracy=1000;
//--- Arrays für Werte von Daten und Preise, die für Setzung und Änderung
//--- der Koordinaten von Bezugspunkte des Preisschilds verwendet werden
    datetime date[];
    double price[];
//--- Speicher reservieren
    ArrayResize(date,bars);
    ArrayResize(price,accuracy);
//--- Füllen die Anordnung von Daten
    ResetLastError();
    if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
    {
        Print("Kann nicht den Wert der Zeit kopieren! Fehlercode = ",GetLastError());
        return;
    }
; }
//--- Füllen den Array der Preise
//--- den Minimal- und Maximalwert der Charts finden
    double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
    double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- Schritte der Preisänderung finden und das Array füllen
    double step=(max_price-min_price)/accuracy;
    for(int i=0;i<accuracy;i++)
        price[i]=min_price+i*step;
//--- Punkte für das Zeichnen des Schilds definieren
    int d=InpDate*(bars-1)/100;
    int p=InpPrice*(accuracy-1)/100;
//--- erstellen das rechte Preisschild auf dem Chart
    if(!ArrowRightPriceCreate(0,InpName,0,date[d],price[p],InpColor,
        InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
    {
        return;
    }

```

```
    ;}
//--- den Chart neu zeichnen und 1 Sekunde warten
    ChartRedraw();
    Sleep(1000);
//--- Jetzt werden wir den Bezugspunkt bewegen
//--- Schleifenzähler
    int v_steps=accuracy*4/5;
//--- bewegen den Bezugspunkt
    for(int i=0;i<v_steps;i++)
    {
        //--- den nächsten Wert nehmen
        if(p>1)
            p-=1;
        //--- den Punkt bewegen
        if(!ArrowRightPriceMove(0,InpName,date[d],price[p]))
            return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
        if(IsStopped())
            return;
        //--- den Chart neu zeichnen
        ChartRedraw();
    }
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//--- Schild aus dem Chart löschen
    ArrowRightPriceDelete(0,InpName);
    ChartRedraw();
//--- 1 Sekunde Verzögerung
    Sleep(1000);
//---
    ;}
```

## OBJ\_ARROW\_BUY

Zeichen "Buy".



### Beispiel

Das folgende Skript erstellt und bewegt auf dem Chart Zeichen "Buy". Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Der Skript zieht Zeichen \"Buy\" im Chartfenster."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input color InpColor=C'3,95,172'; // Farbe des Symbols
//+-----+
//| Erstellt das Zeichen "Buy" |
//+-----+

bool ArrowBuyCreate(const long      chart_ID=0,      // ID des Charts
                   const string    name="ArrowBuy",  // Name des Zeichens
                   const int        sub_window=0,    // Nummer des Fensters
                   datetime         time=0,          // Zeit des Bezugspunkts
                   double           price=0,         // Preis des Bezugspunkts
                   const color      clr=C'3,95,172', // Farbe des Zeichens
                   const ENUM_LINE_STYLE style=STYLE_SOLID, // Liniensstil (für Markie
                   const int        width=1,         // Liniengröße (für Marki
                   const bool        back=false,     // In der Hintergrund
                   const bool        selection=false, // Markieren für Bewegung
```

```

        const bool      hidden=true,          // Verborgen in der Liste
        const long      z_order=0)           // Priorität für Mausclick

    {
//--- Die Koordinaten des Bezugspunkts angeben, wenn sie nicht gesetzt sind
        ChangeArrowEmptyPoint(time,price);
//--- Setzen den Wert des Fehlers zurück
        ResetLastError();
//--- Zeichen erstellen
        if(!ObjectCreate(chart_ID,name,OBJ_ARROW_BUY,sub_window,time,price))
        {
            Print(__FUNCTION__,
                  ": Icon \"Buy\" konnte nicht erstellt werden! Fehlercode = ",GetLastError());
            return(false);
        };
//--- Die Farbe des Zeichens setzen
        ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- Linienstil für Markierung setzen
        ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- Liniengröße für Markierung setzen
        ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
        ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- Aktivieren (true) oder deaktivieren (false) Mausbewegung Modus
        ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
        ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der
        ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- setzen die Priorität für eine Mausclick-Ereignisse auf dem Chart
        ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
        return(true);
    };
//+-----+
//| Bewegt den Bezugspunkt |
//+-----+
bool ArrowBuyMove(const long   chart_ID=0,      // ID des Charts
                  const string name="ArrowBuy", // Objektname
                  datetime    time=0,          // Zeitkoordinate des Bezugspunkts
                  double       price=0)         // Preiskoordinate des Bezugspunkts
{
//--- Wenn die Koordinaten des Punktes nicht angegeben sind, dann verschieben wir es
    if(!time)
        time=TimeCurrent();
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- bewegen den Bezugspunkt
    if(!ObjectMove(chart_ID,name,0,time,price))

```

```

    {
        Print(__FUNCTION__,
            ": der Bezugspunkt konnte nicht bewegt werden! Fehlercode = ", GetLastError());
        return(false);
    }
}

//--- die erfolgreiche Umsetzung
return(true);
};

//+-----+
//| Löscht das Zeichen "Buy" |
//+-----+
bool ArrowBuyDelete(const long   chart_ID=0,      // ID des Charts
                    const string name="ArrowBuy") // Name des Zeichens
{
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Zeichen löschen
    if(!ObjectDelete(chart_ID,name))
    {
        Print(__FUNCTION__,
            ": Zeichen \"Buy\" konnte nicht gelöscht werden! Fehlercode = ", GetLastError());
        return(false);
    }
}

//--- die erfolgreiche Umsetzung
return(true);
};

//+-----+
//| Überprüft den Wert des Bezugspunkts für lehere Werte |
//| setzt Standardwerte |
//+-----+
void ChangeArrowEmptyPoint(datetime &time,double &price)
{
    //--- Wenn Punktzeit nicht angegeben ist, wird es auf dem aktuellen Bar sein
    if(!time)
        time=TimeCurrent();
    //--- Wenn der Preis des Punktes nicht angegeben wird, dann wird es einen Wert von Bid
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
};

//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    datetime date[]; // Array zu speichern Daten von sichtbaren Bars
    double   low[];  // Array zu speichern Low-Preise von sichtbaren Bars
    double   high[]; // Array zu speichern High-Preise von sichtbaren Bars
    //--- Die Anzahl der sichtbaren Bars im Chart-Fenster
    int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);

```

```

//--- Speicher reservieren
ArrayResize(date,bars);
ArrayResize(low,bars);
ArrayResize(high,bars);
//--- Füllen die Anordnung von Daten
ResetLastError();
if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
{
    Print("Kann nicht den Wert der Zeit kopieren! Fehlercode = ",GetLastError());
    return;
};
//--- Füllen die Anordnung von Low-Preise
if(CopyLow(Symbol(),Period(),0,bars,low)==-1)
{
    Print("Kann nicht den Wert der Low-Preise kopieren! Fehlercode = ",GetLastError());
    return;
};
//--- Füllen die Anordnung von High-Preise
if(CopyHigh(Symbol(),Period(),0,bars,high)==-1)
{
    Print("Kann nicht den Wert der High-Preise kopieren! Fehlercode = ",GetLastError());
    return;
};
//--- Erstellen Zeichen "Buy" auf Low für jedes sichtbare Bar
for(int i=0;i<bars;i++)
{
    if(!ArrowBuyCreate(0,"ArrowBuy_"+(string)i,0,date[i],low[i],InpColor))
        return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
if(IsStopped())
    return;
//--- den Chart neu zeichnen
ChartRedraw();
// Verzögerung 0.05 Sekunden
Sleep(50);
};
//--- Bewegen Zeichens "Buy" auf High für jedes sichtbare Bar
for(int i=0;i<bars;i++)
{
    if(!ArrowBuyMove(0,"ArrowBuy_"+(string)i,date[i],high[i]))
        return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
if(IsStopped())
    return;
//--- den Chart neu zeichnen
ChartRedraw();
// Verzögerung 0.05 Sekunden
Sleep(50);
};

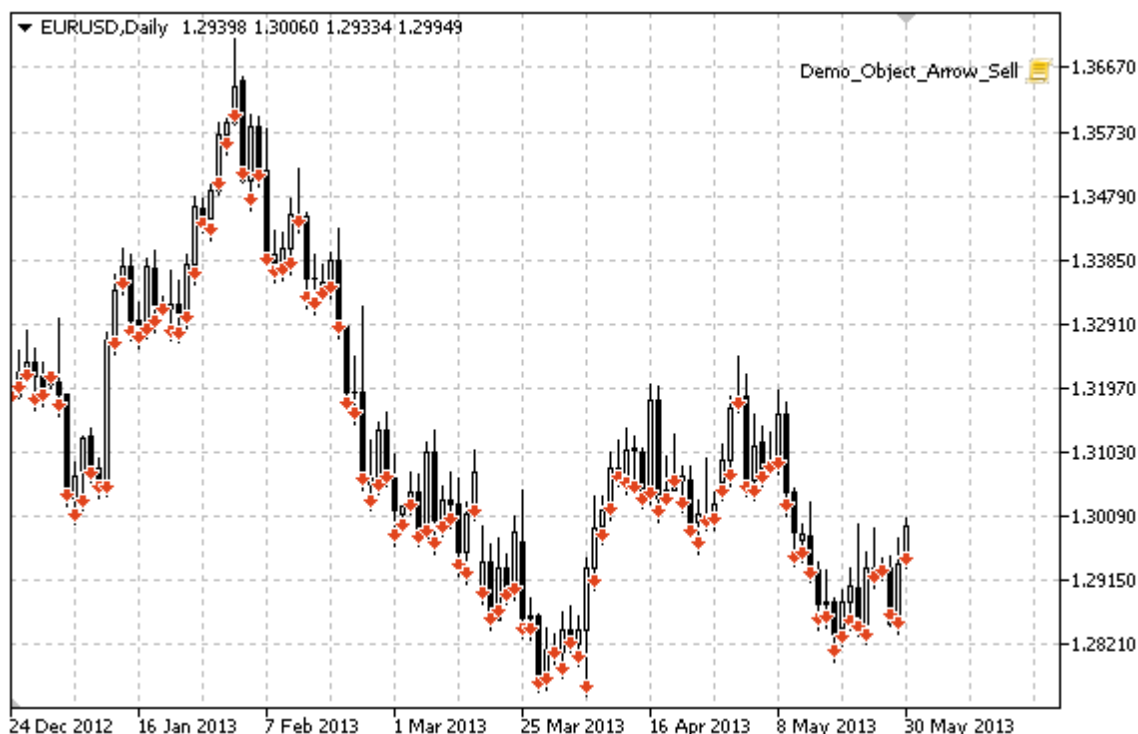
```



```
//--- Zeichen "Buy" löschen
for(int i=0;i<bars;i++)
{
    if(!ArrowBuyDelete(0,"ArrowBuy_"+(string)i))
        return;
    //--- den Chart neu zeichnen
    ChartRedraw();
    // Verzögerung 0.05 Sekunden
    Sleep(50);
};
//---
};
```

## OBJ\_ARROW\_SELL

Zeichen "Sell".



### Beispiel

Das folgende Skript erstellt und bewegt auf dem Chart Zeichen "Sell". Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Der Skript zieht Zeichen \"Sell\" im Chartfenster."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input color InpColor=C'225,68,29'; // Farbe der Zeichen
//+-----+
//| Erstellt das Zeichen "Sell" |
//+-----+

bool ArrowSellCreate(const long      chart_ID=0,          // ID des Charts
                    const string    name="ArrowSell",     // Name der Zeichen
                    const int        sub_window=0,        // Nummer des Unterfensters
                    datetime         time=0,              // Zeit des Bezugspunkts
                    double           price=0,             // Preis des Bezugspunkts
                    const color      clr=C'225,68,29',    // Farbe des Zeichens
                    const ENUM_LINE_STYLE style=STYLE_SOLID, // Liniensstil (zu wählen)
                    const int        width=1,            // Liniengröße (zu wählen)
                    const bool       back=false,         // In der Hintergrund
                    const bool       selection=false,     // Wählen für Bewegung)
```

```

        const bool        hidden=true,           // Verborgen in der Liste der
        const long        z_order=0)           // Priorität für Mausclicks
    {
//--- Die Koordinaten des Bezugspunkts angeben, wenn sie nicht gesetzt sind
        ChangeArrowEmptyPoint(time,price);
//--- Setzen den Wert des Fehlers zurück
        ResetLastError();
//--- Zeichen erstellen
        if(!ObjectCreate(chart_ID,name,OBJ_ARROW_SELL,sub_window,time,price))
        {
            Print(__FUNCTION__,
                ": Kann nicht Zeichen \"Sell\" erstellen! Fehlercode = ",GetLastError());
            return(false);
        };
//--- Die Farbe des Zeichens setzen
        ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- Linienstil für Markierung setzen
        ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- Liniengröße für Markierung setzen
        ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
        ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- Aktivieren (true) oder deaktivieren (false) Mausbewegung Modus
        ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
        ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der
        ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- setzen die Priorität für eine Mausclick-Ereignisse auf dem Chart
        ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
        return(true);
    };
//+-----+
//| Bewegt den Bezugspunkt |
//+-----+
bool ArrowSellMove(const long   chart_ID=0,       // ID des Charts
                  const string name="ArrowSell", // Name des Objekts
                  datetime      time=0,          // Zeitkoordinate des Bezugspunkts
                  double         price=0)         // Preiskoordinate des Bezugspunkts
{
//--- Wenn die Koordinaten des Punktes nicht angegeben sind, dann verschieben wir es
    if(!time)
        time=TimeCurrent();
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- bewegen den Bezugspunkt
    if(!ObjectMove(chart_ID,name,0,time,price))

```

```

    {
        Print(__FUNCTION__,
            ": der Bezugspunkt konnte nicht bewegt werden! Fehlercode = ", GetLastError());
        return(false);
    }
}
//--- die erfolgreiche Umsetzung
return(true);
; }
//+-----+
//| Löscht das Zeichen "Sell" |
//+-----+
bool ArrowSellDelete(const long chart_ID=0, // ID des Charts
                    const string name="ArrowSell") // Name des Zeichens
{
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Zeichen löschen
    if(!ObjectDelete(chart_ID,name))
    {
        Print(__FUNCTION__,
            ": Kann nicht Zeichen \"Sell\" löschen! Fehlercode = ", GetLastError());
        return(false);
    }
}
//--- die erfolgreiche Umsetzung
return(true);
; }
//+-----+
//| Überprüft den Wert des Bezugspunkts für lehre Werte |
//| setzt Standardwerte |
//+-----+
void ChangeArrowEmptyPoint(datetime &time,double &price)
{
    //--- Wenn Punktzeit nicht angegeben ist, wird es auf dem aktuellen Bar sein
    if(!time)
        time=TimeCurrent();
    //--- Wenn der Preis des Punktes nicht angegeben wird, dann wird es einen Wert von Bid
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
}
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    datetime date[]; // Array zu speichern Daten von sichtbaren Bars
    double low[]; // Array zu speichern Low-Priese von sichtbaren Bars
    double high[]; // Array zu speichern High-Preise von sichtbaren Bars
    //--- Die Anzahl der sichtbaren Bars im Chart-Fenster
    int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);

```

```

//--- Speicher reservieren
ArrayResize(date,bars);
ArrayResize(low,bars);
ArrayResize(high,bars);
//--- Füllen die Anordnung von Daten
ResetLastError();
if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
{
    Print("Kann nicht den Wert der Zeit kopieren! Fehlercode = ",GetLastError());
    return;
};
//--- Füllen die Anordnung von Low-Preise
if(CopyLow(Symbol(),Period(),0,bars,low)==-1)
{
    Print("Kann nicht den Wert der Low-Preise kopieren! Fehlercode = ",GetLastError());
    return;
};
//--- Füllen die Anordnung von High-Preise
if(CopyHigh(Symbol(),Period(),0,bars,high)==-1)
{
    Print("Kann nicht den Wert der High-Preise kopieren! Fehlercode = ",GetLastError());
    return;
};
//--- Erstellen Icons "Sell" auf High für jedes sichtbare Bar
for(int i=0;i<bars;i++)
{
    if(!ArrowSellCreate(0,"ArrowSell_"+(string)i,0,date[i],high[i],InpColor))
        return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
if(IsStopped())
    return;
//--- den Chart neu zeichnen
ChartRedraw();
// Verzögerung 0.05 Sekunden
Sleep(50);
};
//--- Bewegen Icons "Sell" auf Low für jedes sichtbare Bar
for(int i=0;i<bars;i++)
{
    if(!ArrowSellMove(0,"ArrowSell_"+(string)i,date[i],low[i]))
        return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
if(IsStopped())
    return;
//--- den Chart neu zeichnen
ChartRedraw();
// Verzögerung 0.05 Sekunden
Sleep(50);
};

```

```
//--- Zeichen "Sell" löschen
for(int i=0;i<bars;i++)
{
    if(!ArrowSellDelete(0,"ArrowSell_"+(string)i))
        return;
    //--- den Chart neu zeichnen
    ChartRedraw();
    // Verzögerung 0.05 Sekunden
    Sleep(50);
};
//---
};
```

## OBJ\_ARROW

Zeichen "Pfeil".



### Hinweis

Die Position des Bezugspunktes relativ zum Zeichen kann aus der Aufzählung [ENUM\\_ARROW\\_ANCHOR](#) gewählt werden.

Größe Pfeile (mehr als 5) können nur durch Setzen der entsprechenden Eigenschaft [OBJPROP\\_WIDTH](#) beim Schreiben von Code in MetaEditor erstellt werden.

Die gewünschte Art der Pfeile kann durch Einstellung einer der Codes von Symbole von Font [Wingdings](#) ausgewählt werden.

### Beispiel

Das folgende Skript erstellt Objekt "Pfeil" auf dem Chart und ändert seinen Typ. Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Das Skript erstellt einen zufälligen Pfeil im Chartfenster."
#property description "Koordinate des Bezugspunktes wird in Prozent"
#property description "von der Größe des Chartfensters eingegeben."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="Arrow";           // Name des Pfeils
input int         InpDate=50;                // Datum des Bezugspunktes in %
```

```

input int                InpPrice=50;           // Preis des Bezugspunktes in %
input ENUM_ARROW_ANCHOR InpAnchor=ANCHOR_TOP;   // Bezugsmethode
input color              InpColor=clrDodgerBlue; // Farbe des Pfeils
input ENUM_LINE_STYLE    InpStyle=STYLE_SOLID;   // Stil der Einfassungslinie
input int                InpWidth=10;           // Größe des Pfeils
input bool               InpBack=false;         // Pfeil im Hintergrund
input bool               InpSelection=false;     // Wählen zu bewegen
input bool               InpHidden=true;        // Ausgeblendet in der Liste der Objekte
input long               InpZOrder=0;           // Priorität auf Mausklick

//+-----+
//| Erstellt einen Pfeil |
//+-----+
bool ArrowCreate(const long      chart_ID=0,      // ID des Charts
                 const string   name="Arrow",    // Name des Pfeils
                 const int      sub_window=0,    // Nummer des Unterfensters
                 datetime        time=0,         // Zeit des Bezugspunktes
                 double          price=0,        // Preis des Bezugspunktes
                 const uchar     arrow_code=252,  // Code des Pfeils
                 const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // Position des Bezugspunktes
                 const color     clr=clrRed,     // Farbe des Pfeils
                 const ENUM_LINE_STYLE style=STYLE_SOLID, // Stil der Einfassungslinie
                 const int       width=3,        // Größe des Pfeils
                 const bool      back=false,     // Im Hintergrund
                 const bool      selection=true,  // Wählen zu bewegen
                 const bool      hidden=true,    // Verborgen in der Liste der Objekte
                 const long      z_order=0)      // Priorität auf Mausklick
{
    //--- Die Koordinaten des Bezugspunktes angeben, wenn sie nicht gesetzt sind
    ChangeArrowEmptyPoint(time,price);
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Pfeil erstellen
    if(!ObjectCreate(chart_ID,name,OBJ_ARROW,sub_window,time,price))
    {
        Print(__FUNCTION__,
              ": Pfeil konnte nicht erstellt werden! Fehlercode = ",GetLastError());
        return(false);
    }
    //--- Code des Pfeils setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_ARROWCODE,arrow_code);
    //--- die Bindungsmethode setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
    //--- Farbe des Pfeils setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
    //--- Stil der Einfassungslinie setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
    //--- Größe des Pfeils setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
    //--- Im Vordergrund (false) oder Hintergrund (true) anzeigen

```



```

    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- Aktivieren (true) oder deaktivieren (false) Mausbewegung Modus
//--- Wenn Sie ein grafisches Objekt durch Funktion ObjectCreate erstellen, das Objekt
//--- kann nicht ausgewählt und verschoben werden. Innerhalb dieser Methode ist der St
//--- true, was Sie das Objekt auswählen und verschieben erlaubt
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der
    ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- setzen die Priorität für eine Mausklick-Ereignisse auf dem Chart
    ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
    return(true);
; }
//+-----+
//| Bewegt den Bezugspunkt |
//+-----+
bool ArrowMove(const long   chart_ID=0,    // ID des Charts
               const string name="Arrow",  // Name des Objekts
               datetime     time=0,        // Zeitkoordinate des Bezugspunkts
               double        price=0)      // Preiskoordinate des Bezugspunkts
{
//--- Wenn die Koordinaten des Punktes nicht angegeben sind, dann verschieben wir es
    if(!time)
        time=TimeCurrent();
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- bewegen den Bezugspunkt
    if(!ObjectMove(chart_ID,name,0,time,price))
    {
        Print(__FUNCTION__,
              ": der Bezugspunkt konnte nicht bewegt werden! Fehlercode = ",GetLastError());
        return(false);
    }
; }
//--- die erfolgreiche Umsetzung
    return(true);
; }
//+-----+
//| Ändert Code des Pfeils |
//+-----+
bool ArrowCodeChange(const long   chart_ID=0,    // ID des Charts
                    const string name="Arrow",  // Name des Objekts
                    const uchar  code=252)     // Code des Pfeils
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Code des Pfeils ändern

```

```

    if(!ObjectSetInteger(chart_ID,name,OBJPROP_ARROWCODE,code))
    {
        Print(__FUNCTION__,
            ": Code des Pfeils konnte nicht geändert werden! Fehlercode = ",GetLastError());
        return(false);
    }
; }

//--- die erfolgreiche Umsetzung
return(true);
; }

//+-----+
//| Ändert Bindungsmethode |
//+-----+

bool ArrowAnchorChange(const long      chart_ID=0,      // ID des Charts
                      const string    name="Arrow",    // Name des Objekts
                      const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // Bindungsmethode
{
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Bindungsmethode ändern
    if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
    {
        Print(__FUNCTION__,
            ": Bindungsmethode konnte nicht geändert werden! Fehlercode = ",GetLastError());
        return(false);
    }
; }

//--- die erfolgreiche Umsetzung
return(true);
; }

//+-----+
//| Löscht den Pfeil |
//+-----+

bool ArrowDelete(const long chart_ID=0, // ID des Charts
                 const string name="Arrow") // Name des Pfeils
{
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Pfeil löschen
    if(!ObjectDelete(chart_ID,name))
    {
        Print(__FUNCTION__,
            ": Pfeil konnte nicht gelöscht werden! Fehlercode = ",GetLastError());
        return(false);
    }
; }

//--- die erfolgreiche Umsetzung
return(true);
; }

//+-----+
//| Überprüft den Wert des Bezugspunkts für lehere Werte |
//| setzt Standardwerte |

```

```
//+-----+
void ChangeArrowEmptyPoint(datetime &time,double &price)
{
//--- Wenn Punktzeit nicht angegeben ist, wird es auf dem aktuellen Bar sein
    if(!time)
        time=TimeCurrent();
//--- Wenn der Preis des Punktes nicht angegeben wird, dann wird es einen Wert von Bid
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
};
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//--- Überprüfen die Richtigkeit der Eingabeparameter
    if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
    {
        Print("Fehler! Ungültige Eingabeparameter! ");
        return;
    };
//--- Die Anzahl der sichtbaren Bars im Chart-Fenster
    int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- Größe des Arrays price
    int accuracy=1000;
//--- Arrays für Werte von Daten und Preise, die für Setzung und Änderung
//--- der Koordinaten von Ankerpunkte verwendet werden
    datetime date[];
    double price[];
//--- Speicher reservieren
    ArrayResize(date,bars);
    ArrayResize(price,accuracy);
//--- Füllen die Anordnung von Daten
    ResetLastError();
    if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
    {
        Print("Kann nicht den Wert der Zeit kopieren! Fehlercode = ",GetLastError());
        return;
    };
//--- Füllen den Array der Preise
//--- den Minimal- und Maximalwert der Charts finden
    double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
    double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- Schritte der Preisänderung finden und das Array füllen
    double step=(max_price-min_price)/accuracy;
    for(int i=0;i<accuracy;i++)
        price[i]=min_price+i*step;
//--- Punkte für das Zeichnen des Pfeils definieren
    int d=InpDate*(bars-1)/100;
```

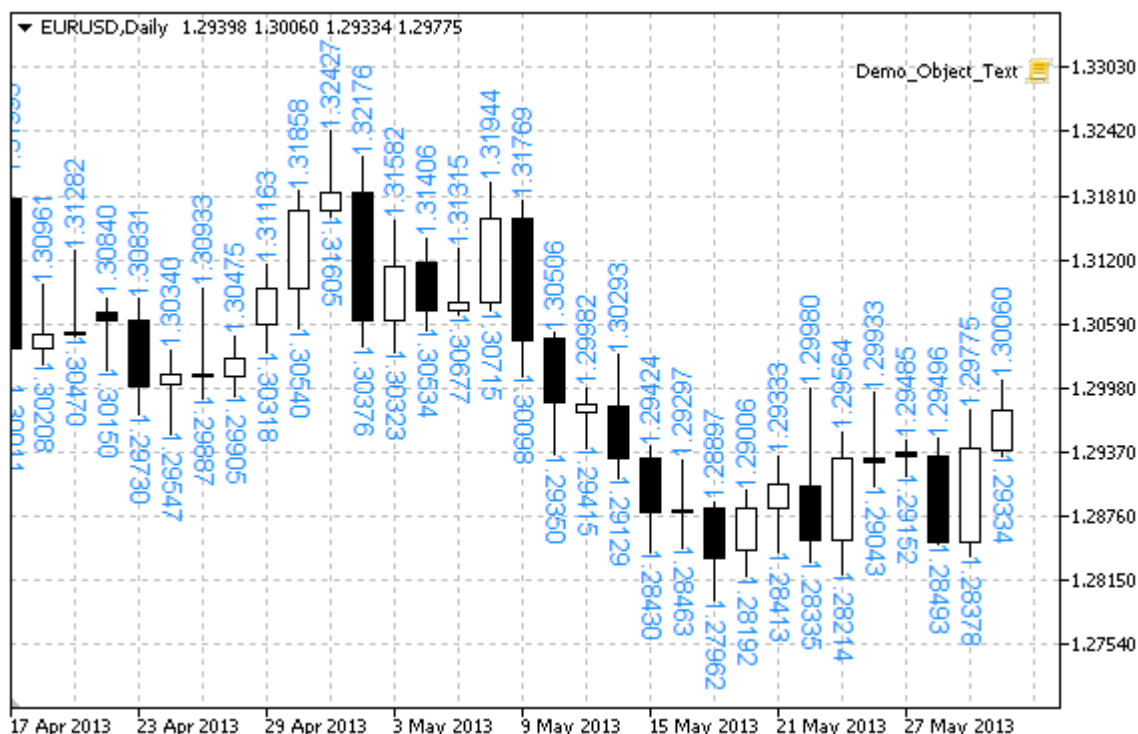
```

    int p=InpPrice*(accuracy-1)/100;
    ///--- erstellen einen Pfeil auf dem Chart
    if(!ArrowCreate(0,InpName,0,date[d],price[p],32,InpAnchor,InpColor,
        InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
    {
        return;
    }
    ///--- den Chart neu zeichnen
    ChartRedraw();
    ///--- Alle Varianten von Erstellung der Pfeile in Zyklus betrachten
    for(int i=33;i<256;i++)
    {
        if(!ArrowCodeChange(0,InpName,(uchar)i))
            return;
        ///--- Überprüfen die Fakten von Zwangsabschaltung der Skript
        if(IsStopped())
            return;
        ///--- den Chart neu zeichnen
        ChartRedraw();
        /// Verzögerung in einer halben Sekunde
        Sleep(500);
    }
    ///--- 1 Sekunde Verzögerung
    Sleep(1000);
    ///--- Pfeil aus dem Chart löschen
    ArrowDelete(0,InpName);
    ChartRedraw();
    ///--- 1 Sekunde Verzögerung
    Sleep(1000);
    ///---
    ;}

```

## OBJ\_TEXT

Objekt "Text".



### Hinweis

Sie können die Position des Ankerpunkts relativ zum Text aus der Aufzählung [ENUM\\_ANCHOR\\_POINT](#) wählen. Sie können auch den Textwinkel durch die Eigenschaft [OBJPROP\\_ANGLE](#) geändert werden.

### Beispiel

Das folgende Skript erstellt auf dem Chart einige Objekte "Text". Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Das Skript erstellt ein graphisches Objekt \"Text\"."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpFont="Arial";           // Schrift
input int         InpFontSize=10;            // Schriftgröße
input color       InpColor=clrRed;           // Farbe
input double      InpAngle=90.0;             // Der Winkel in Grad
input ENUM_ANCHOR_POINT InpAnchor=ANCHOR_BOTTOM; // Bindungsmethode
input bool        InpBack=false;             // Objekt im Hintergrund
input bool        InpSelection=false;        // Wählen zu bewegen
input bool        InpHidden=true;           // Ausblenden in der Liste der Objekten
input long        InpZOrder=0;              // Priorität auf Mausklick
```

```
//+-----+
//| Erstellt das Objekt "Text" |
//+-----+
bool TextCreate(const long      chart_ID=0,          // ID des Charts
               const string    name="Text",         // Objektname
               const int       sub_window=0,        // Nummer des Unterfensters
               datetime        time=0,              // Zeit des Ankerpunkts
               double          price=0,             // Preis des Ankerpunkts
               const string    text="Text",         // Der Text
               const string    font="Arial",        // Schrift
               const int       font_size=10,        // Schriftgröße
               const color     clr=clrRed,          // Farbe
               const double    angle=0.0,          // Text Winkel
               const ENUM_ANCHOR_POINT anchor=ANCHOR_LEFT_UPPER, // Bindungsmethode
               const bool     back=false,          // Im Hintergrund
               const bool     selection=false,      // Wählen um zu bewegen
               const bool     hidden=true,         // Ausgeblendet in der Liste
               const long      z_order=0)          // Priorität auf Mausaktionen

{
    //--- Die Koordinaten des Bezugspunkts angeben, wenn sie nicht gesetzt sind
    ChangeTextEmptyPoint(time,price);
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Erstellen wir das Objekt "Text"
    if(!ObjectCreate(chart_ID,name,OBJ_TEXT,sub_window,time,price))
    {
        Print(__FUNCTION__,
              ": Objekt \"Text\" konnte nicht erstellt werden! Fehlercode = ",GetLastError());
        return(false);
    }
    //--- den Text setzen
    ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
    //--- Textschrift setzen
    ObjectSetString(chart_ID,name,OBJPROP_FONT,font);
    //--- Schriftgröße setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
    //--- Text-Winkel angeben
    ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);
    //--- die Bindungsmethode setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
    //--- Farbe setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
    //--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
    //--- aktivieren (true) oder deaktivieren (false) Bewegung des Objekts mit dem Maus
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
    //--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der Liste
    ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
}
```

```

//--- setzen die Priorität für eine Mausklick-Ereignisse auf dem Chart
    ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
    return(true);
; }

//+-----+
//| Bewegt den Bezugspunkt |
//+-----+
bool TextMove(const long   chart_ID=0, // ID des Charts
              const string name="Text", // Objektname
              datetime     time=0,      // Zeitkoordinate des Punktes
              double        price=0)    // Preiskoordinate des Punktes
{
//--- Wenn die Koordinaten des Punktes nicht angegeben sind, dann verschieben wir es
    if(!time)
        time=TimeCurrent();
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- bewegen den Bezugspunkt
    if(!ObjectMove(chart_ID,name,0,time,price))
    {
        Print(__FUNCTION__,
              ": der Bezugspunkt konnte nicht bewegt werden! Fehlercode = ",GetLastError());
        return(false);
    }
; }

//--- die erfolgreiche Umsetzung
    return(true);
; }

//+-----+
//| Ändert den Text des Objektes |
//+-----+
bool TextChange(const long   chart_ID=0, // ID des Charts
                const string name="Text", // Objektname
                const string text="Text") // Text
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- ändern wir den Text des Objektes
    if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
    {
        Print(__FUNCTION__,
              ": Konnte nicht den Text ändern! Fehlercode = ",GetLastError());
        return(false);
    }
; }

//--- die erfolgreiche Umsetzung
    return(true);
; }

```

```

//+-----+
//| Löscht das Objekt "Text" |
//+-----+
bool TextDelete(const long chart_ID=0, // ID des Charts
                const string name="Text") // Objektname
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- das Objekt löschen
    if(!ObjectDelete(chart_ID,name))
    {
        Print(__FUNCTION__,
              ": Objekt \"Text\" konnte nicht gelöscht werden! Fehlercode = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Überprüft den Wert des Bezugspunkts für leere Werte |
//| setzt Standardwerte |
//+-----+
void ChangeTextEmptyPoint(datetime &time,double &price)
{
//--- Wenn Punktzeit nicht angegeben ist, wird es auf dem aktuellen Bar sein
    if(!time)
        time=TimeCurrent();
//--- Wenn der Preis des Punktes nicht angegeben wird, dann wird es einen Wert von Bid
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
};
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    datetime date[]; // Array zu speichern Daten von sichtbaren Bars
    double low[]; // Array zu speichern Low-Preise von sichtbaren Bars
    double high[]; // Array zu speichern High-Preise von sichtbaren Bars
//--- Die Anzahl der sichtbaren Bars im Chart-Fenster
    int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- Speicher reservieren
    ArrayResize(date,bars);
    ArrayResize(low,bars);
    ArrayResize(high,bars);
//--- Füllen die Anordnung von Daten
    ResetLastError();
    if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
    {

```



```

        Print("Kann nicht den Wert der Zeit kopieren! Fehlercode = ",GetLastError());
        return;
    };
}

//--- Füllen die Anordnung von Low-Preise
if(CopyLow(Symbol(),Period(),0,bars,low)==-1)
{
    Print("Kann nicht den Wert der Low-Preise kopieren! Fehlercode = ",GetLastError());
    return;
};

//--- Füllen die Anordnung von High-Preise
if(CopyHigh(Symbol(),Period(),0,bars,high)==-1)
{
    Print("Kann nicht den Wert der High-Preise kopieren! Fehlercode = ",GetLastError());
    return;
};

//--- Definieren wir, wie oft die Inschriften zu Tun
int scale=(int)ChartGetInteger(0,CHART_SCALE);
//--- Schritt definieren
int step=1;
switch(scale)
{
    case 0:
        step=12;
        break;
    case 1:
        step=6;
        break;
    case 2:
        step=4;
        break;
    case 3:
        step=2;
        break;
};

//--- Text für die Werte von High und Low der Baren erstellen (mit Zwischenräume)
for(int i=0;i<bars;i+=step)
{
    //--- Texte erstellen
    if(!TextCreate(0,"TextHigh_"+(string)i,0,date[i],high[i],DoubleToString(high[i],
        InpColor,InpAngle,InpAnchor,InpBack,InpSelection,InpHidden,InpZOrder))
    {
        return;
    };
    if(!TextCreate(0,"TextLow_"+(string)i,0,date[i],low[i],DoubleToString(low[i],5),
        InpColor,-InpAngle,InpAnchor,InpBack,InpSelection,InpHidden,InpZOrder))
    {
        return;
    };
};

//--- Überprüfen die Fakten von Zwangsabschaltung der Skript

```

```
        if(IsStopped())
            return;
        //--- den Chart neu zeichnen
        ChartRedraw();
        // Verzögerung 0.05 Sekunden
        Sleep(50);
    };
//--- Halb-Sekunden-Verzögerung
Sleep(500);
//--- Texte löschen
for(int i=0;i<bars;i+=step)
{
    if(!TextDelete(0,"TextHigh_"+(string)i))
        return;
    if(!TextDelete(0,"TextLow_"+(string)i))
        return;
    //--- den Chart neu zeichnen
    ChartRedraw();
    // Verzögerung 0.05 Sekunden
    Sleep(50);
};
//---
};
```

## OBJ\_LABEL

Objekt "Text-Label."



### Hinweis

Die Position des Bezugspunktes relativ zum Zeichen kann aus der Aufzählung [ENUM\\_ANCHOR\\_POINT](#) gewählt werden. Koordinaten des Bezugspunktes werden in Pixel angegeben.

Sie können den Winkel des Anbindepunktes des Labels durch Aufzählung [ENUM\\_BASE\\_CORNER](#) wählen.

### Beispiel

Das folgende Skript erstellt und bewegt auf dem Chart Objekt "Text-Label". Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Das Skript erstellt ein graphisches Objekt \"Text-Label\"."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="Label";           // Name des Labels
input int         InpX=150;                  // X Abstand
input int         InpY=150;                  // Y Abstand
input string      InpFont="Arial";           // Schrift
input int         InpFontSize=14;            // Schriftgröße
input color       InpColor=clrRed;           // Farbe
input double      InpAngle=0.0;              // Der Winkel in Grad
```

```

input ENUM_ANCHOR_POINT InpAnchor=ANCHOR_CENTER; // Bindungsmethode
input bool               InpBack=false;           // Objekt im Hintergrund
input bool               InpSelection=true;        // Wählen zu bewegen
input bool               InpHidden=true;          // Ausblenden in der Liste der Objekten
input long               InpZOrder=0;             // Priorität auf Mausklick

//+-----+
//| Erstellt ein Text-Label                                     |
//+-----+
bool LabelCreate(const long      chart_ID=0,        // ID des Charts
                 const string   name="Label",      // Name des Labels
                 const int      sub_window=0,      // Nummer des Unter
                 const int      x=0,               // X-Koordinate
                 const int      y=0,               // Y-Koordinate
                 const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER, // Winkel des Chart
                 const string   text="Label",      // Text
                 const string   font="Arial",      // Schrift
                 const int      font_size=10,      // Schriftgröße
                 const color     clr=clrRed,        // Farbe
                 const double    angle=0.0,        // Text Winkel
                 const ENUM_ANCHOR_POINT anchor=ANCHOR_LEFT_UPPER, // Bindungsmethode
                 const bool      back=false,        // Im Hintergrund
                 const bool      selection=false,   // Wählen um zu bev
                 const bool      hidden=true,       // Ausgeblendet in
                 const long      z_order=0)         // Priorität auf Ma

{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- ein Text-Label erstellen
    if(!ObjectCreate(chart_ID,name,OBJ_LABEL,sub_window,0,0))
    {
        Print(__FUNCTION__,
              ": Text-Label konnte nicht erstellt werden! Fehlercode = ",GetLastError());
        return(false);
    };
//--- Die Koordinaten des Schilds setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
    ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- wählen die Ecke des Charts, relativ zu der die Punktkoordinaten eingegeben werde
    ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- den Text setzen
    ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- Textschrift setzen
    ObjectSetString(chart_ID,name,OBJPROP_FONT,font);
//--- Schriftgröße setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
//--- Text-Winkel angeben
    ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);
//--- die Bindungsmethode setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);

```

```

//--- Farbe setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- Aktivieren (true) oder deaktivieren (false) Mausbewegung Modus
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der
    ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- setzen die Priorität für eine Mausklick-Ereignisse auf dem Chart
    ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
    return(true);
; }

//+-----+
//| Bewegt ein Text-Label |
//+-----+

bool LabelMove(const long   chart_ID=0,    // ID des Charts
               const string name="Label",  // Name des Labels
               const int    x=0,           // X-Koordinate
               const int    y=0)           // Y-Koordinate
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- ein Text-Label bewegen
    if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
    {
        Print(__FUNCTION__,
              ": Kann nicht X-Koordinate des Labels bewegen! Fehlercode = ",GetLastError());
        return(false);
    }
; }
    if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
    {
        Print(__FUNCTION__,
              ": Kann nicht Y-Koordinate des Labels bewegen! Fehlercode = ",GetLastError());
        return(false);
    }
; }
//--- die erfolgreiche Umsetzung
    return(true);
; }

//+-----+
//| Ändert Winkel des Charts das Label zu binden |
//+-----+

bool LabelChangeCorner(const long   chart_ID=0,    // ID des Charts
                      const string name="Label",  // Name des Labels
                      const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER) // Winkel des Labels
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();

```

```

//--- Winkel der Bindung ändern
if(!ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner))
{
    Print(__FUNCTION__,
        ": Kann den Winkel der Bindung nicht ändern! Fehlercode = ",GetLastError())
    return(false);
};
//--- die erfolgreiche Umsetzung
return(true);
};
//+-----+
//| Ändert den Text des Objektes |
//+-----+
bool LabelTextChange(const long   chart_ID=0,    // ID des Charts
                    const string name="Label",  // Objektname
                    const string text="Text")    // Text
{
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- ändern wir den Text des Objektes
    if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
    {
        Print(__FUNCTION__,
            ": Konnte nicht den Text ändern! Fehlercode = ",GetLastError());
        return(false);
    };
    //--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Löscht ein Text-Label |
//+-----+
bool LabelDelete(const long   chart_ID=0,    // ID des Charts
                const string name="Label")  // Name des Labels
{
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- das Schild löschen
    if(!ObjectDelete(chart_ID,name))
    {
        Print(__FUNCTION__,
            ": Text-Label konnte nicht gelöscht werden! Fehlercode = ",GetLastError())
        return(false);
    };
    //--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Script program start function |

```

```
//+-----+
void OnStart()
{
    //--- die Koordinaten des Labels in lokalen Variablen speichern
    int x=InpX;
    int y=InpY;
    //--- Größe des Chartfensters
    long x_distance;
    long y_distance;
    //--- Größe des Chartfensters definieren
    if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
    {
        Print("Konnte nicht Breite des Charts bekommen! Fehlercode = ",GetLastError());
        return;
    }
    if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
    {
        Print("Konnte nicht Höhe des Charts bekommen! Fehlercode = ",GetLastError());
        return;
    }
    //--- Überprüfen die Richtigkeit der Eingabeparameter
    if(InpX<0 || InpX>x_distance-1 || InpY<0 || InpY>y_distance-1)
    {
        Print("Fehler! Ungültige Eingabeparameter! ");
        return;
    }
    //--- den ersten Text für das Label vorbereiten
    string text;
    StringConcatenate(text,"Linke obere Ecke: ",x,"",y);
    //--- ein Text-Label auf dem Chart erstellen
    if(!LabelCreate(0,InpName,0,InpX,InpY,CORNER_LEFT_UPPER,text,InpFont,InpFontSize,
        InpColor,InpAngle,InpAnchor,InpBack,InpSelection,InpHidden,InpZOrder))
    {
        return;
    }
    //--- Den Chart neu zeichnen und halbe Sekunden warten
    ChartRedraw();
    Sleep(500);
    //--- bewegen das Label und ändern seinen Text
    //--- Anzahl der Iterationen für Achsen
    int h_steps=(int)(x_distance/2-InpX);
    int v_steps=(int)(y_distance/2-InpY);
    //--- Text-Label nach unten bewegen
    for(int i=0;i<v_steps;i++)
    {
        //--- Koordinate ändern
        y+=2;
        //--- bewegen das Label und ändern seinen Text
        MoveAndTextChange(x,y,"Linke obere Ecke: ");
    }
}
```

```

    ;}
//--- Halb-Sekunden-Verzögerung
    Sleep(500);
//--- Text-Label nach rechts bewegen
    for(int i=0;i<h_steps;i++)
    {
        //--- Koordinate ändern
        x+=2;
        //--- bewegen das Label und ändern seinen Text
        MoveAndTextChange(x,y,"Linke obere Ecke: ");
    }
//--- Halb-Sekunden-Verzögerung
    Sleep(500);
//--- Text-Label nach oben bewegen
    for(int i=0;i<v_steps;i++)
    {
        //--- Koordinate ändern
        y-=2;
        //--- bewegen das Label und ändern seinen Text
        MoveAndTextChange(x,y,"Linke obere Ecke: ");
    }
//--- Halb-Sekunden-Verzögerung
    Sleep(500);
//--- Text-Label nach links bewegen
    for(int i=0;i<h_steps;i++)
    {
        //--- Koordinate ändern
        x-=2;
        //--- bewegen das Label und ändern seinen Text
        MoveAndTextChange(x,y,"Linke obere Ecke: ");
    }
//--- Halb-Sekunden-Verzögerung
    Sleep(500);
//--- nun bewegen wir den Punkt (Winkel der Bindung ändern)
//--- in den unteren linken Winkel bewegen
    if(!LabelChangeCorner(0,InpName,CORNER_LEFT_LOWER))
        return;
//--- ändern wir den Text des Labels
    StringConcatenate(text,"Linke untere Ecke: ",x," ",y);
    if(!LabelTextChange(0,InpName,text))
        return;
//--- Den Chart neu zeichnen und zwei Sekunden warten
    ChartRedraw();
    Sleep(2000);
//--- in den unteren rechten Winkel bewegen
    if(!LabelChangeCorner(0,InpName,CORNER_RIGHT_LOWER))
        return;
//--- ändern wir den Text des Labels
    StringConcatenate(text,"Rechte untere Ecke: ",x," ",y);

```



```

    if(!LabelTextChange(0, InpName, text))
        return;
//--- Den Chart neu zeichnen und zwei Sekunden warten
    ChartRedraw();
    Sleep(2000);
//--- in den oberen rechten Winkel bewegen
    if(!LabelChangeCorner(0, InpName, CORNER_RIGHT_UPPER))
        return;
//--- ändern wir den Text des Labels
    StringConcatenate(text, "Rechte obere Ecke: ", x, ", ", y);
    if(!LabelTextChange(0, InpName, text))
        return;
//--- Den Chart neu zeichnen und zwei Sekunden warten
    ChartRedraw();
    Sleep(2000);
//--- in den oberen linken Winkel bewegen
    if(!LabelChangeCorner(0, InpName, CORNER_LEFT_UPPER))
        return;
//--- ändern wir den Text des Labels
    StringConcatenate(text, "Linke obere Ecke: ", x, ", ", y);
    if(!LabelTextChange(0, InpName, text))
        return;
//--- Den Chart neu zeichnen und zwei Sekunden warten
    ChartRedraw();
    Sleep(2000);
//--- das Schild löschen
    LabelDelete(0, InpName);
//--- Den Chart neu zeichnen und halbe Sekunden warten
    ChartRedraw();
    Sleep(500);
//---
; }
//+-----+
//| Die Funktion bewegt das Objekt und ändert seinen Text |
//+-----+
bool MoveAndTextChange(const int x, const int y, string text)
{
//--- Label bewegen
    if(!LabelMove(0, InpName, x, y))
        return(false);
//--- ändern wir den Text des Labels
    StringConcatenate(text, text, x, ", ", y);
    if(!LabelTextChange(0, InpName, text))
        return(false);
//--- Überprüfen die Fakten der Zwangsabschaltung von Skript
    if(IsStopped())
        return(false);
//--- den Chart neu zeichnen
    ChartRedraw();

```

```
//--- Verzögerung von 0.01 Sekunden  
    Sleep(10);  
//--- Funktion beendet  
    return(true);  
; }
```

## OBJ\_BUTTON

Objekt "Schaltfläche".



### Hinweis

Koordinaten des Ankerpunktes werden in Pixeln angegeben. Sie können den Winkel des Ankerpunktes der Schaltfläche durch Aufzählung [ENUM\\_BASE\\_CORNER](#) auswählen.

### Beispiel

Das folgende Skript erstellt und bewegt auf dem Chart Objekt "Schaltfläche". Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Das Skript erstellt eine Schaltfläche auf dem Chart."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="Button";           // Schaltflächenname
input ENUM_BASE_CORNER InpCorner=CORNER_LEFT_UPPER; // Winkel des Charts zu binden
input string      InpFont="Arial";           // Schrift
input int         InpFontSize=14;            // Schriftgröße
input color       InpColor=clrBlack;         // Textfarbe
input color       InpBackColor=C'236,233,216'; // Hintergrundfarbe
input color       InpBorderColor=clrNONE;    // Rahmenfarbe
input bool        InpState=false;            // Gedrückt/Ungedrückt
input bool        InpBack=false;            // Objekt im Hintergrund
input bool        InpSelection=false;        // Wählen zu bewegen
```

```

input bool      InpHidden=true;           // Ausgeblendet in der Liste der C
input long      InpZOrder=0;              // Priorität auf Mausklick
//+-----+
//| Erstellt eine Taste                      |
//+-----+
bool ButtonCreate(const long      chart_ID=0,           // ID des Charts
                  const string    name="Button",       // Schaltflächennamen
                  const int       sub_window=0,        // Nummer des Unterfensters
                  const int       x=0,                 // X-Koordinate
                  const int       y=0,                 // Y-Koordinate
                  const int       width=50,            // Breite der Schaltfläche
                  const int       height=18,           // Höhe der Schaltfläche
                  const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER, // Winkel des Charts
                  const string    text="Button",      // Text
                  const string    font="Arial",       // Schrift
                  const int       font_size=10,        // Schriftgröße
                  const color      clr=clrBlack,       // Textfarbe
                  const color      back_clr=C'236,233,216', // Hintergrundfarbe
                  const color      border_clr=clrNONE, // Rahmenfarbe
                  const bool       state=false,        // Gedrückt/Ungedrückt
                  const bool       back=false,         // Im Hintergrund
                  const bool       selection=false,     // Auswählen um zu klicken
                  const bool       hidden=true,        // Ausgeblendet in der Liste der Charts
                  const long       z_order=0)           // Priorität auf Mausklick
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- eine Schaltfläche erstellen
    if(!ObjectCreate(chart_ID,name,OBJ_BUTTON,sub_window,0,0))
    {
        Print(__FUNCTION__,
              ": Konnte nicht Schaltfläche erstellen! Fehlercode = ",GetLastError());
        return(false);
    };
//--- Schaltflächenkoordinaten setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
    ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- Schaltflächengröße setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
    ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- wählen die Ecke des Charts, relativ zu der die Punktkoordinaten eingegeben werden
    ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- den Text setzen
    ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- Textschrift setzen
    ObjectSetString(chart_ID,name,OBJPROP_FONT,font);
//--- Schriftgröße setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
//--- Textfarbe setzen

```

```

    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- Hintergrundfarbe setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_BGCOLOR,back_clr);
//--- Rahmenfarbe setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_COLOR,border_clr);
//--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- set button state
    ObjectSetInteger(chart_ID,name,OBJPROP_STATE,state);
//--- aktivieren (true) oder deaktivieren (false) Bewegung der Schaltfläche mit dem Ma
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der
    ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- setzen die Priorität für eine Mausklick-Ereignisse auf dem Chart
    ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
    return(true);
; }
//+-----+
//| Bewegt die Schaltfläche |
//+-----+
bool ButtonMove(const long   chart_ID=0,    // ID des Charts
               const string name="Button", // Name der Schaltfläche
               const int    x=0,           // X Koordinate
               const int    y=0)           // Y Koordinate
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Schaltfläche bewegen
    if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
    {
        Print(__FUNCTION__,
              ": Kann nicht X-Koordinate der Schaltfläche bewegen! Fehlercode = ",GetLastError());
        return(false);
    ; }
    if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
    {
        Print(__FUNCTION__,
              ": Kann nicht Y-Koordinate der Schaltfläche bewegen! Fehlercode = ",GetLastError());
        return(false);
    ; }
//--- die erfolgreiche Umsetzung
    return(true);
; }
//+-----+
//| Ändert die Größe einer Schaltfläche |
//+-----+
bool ButtonChangeSize(const long   chart_ID=0,    // ID des Charts

```

```

        const string name="Button", // Name der Schaltfläche
        const int   width=50,       // Breite der Schaltfläche
        const int   height=18)      // Höhe der Schaltfläche

    {
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Größe der Schaltfläche ändern
    if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
    {
        Print(__FUNCTION__,
              ": Konnte nicht die Breite der Schaltfläche ändern! Fehlercode = ",GetLastError());
        return(false);
    };
    if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
    {
        Print(__FUNCTION__,
              ": Konnte nicht die Höhe der Schaltfläche ändern! Fehlercode = ",GetLastError());
        return(false);
    };
//--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Ändert Winkel des Charts die Taste zu binden |
//+-----+
bool ButtonChangeCorner(const long   chart_ID=0,           // ID des Charts
                        const string name="Button",        // Name der Schaltfläche
                        const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER) // Winkel der Bindung
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Winkel der Bindung ändern
    if(!ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner))
    {
        Print(__FUNCTION__,
              ": Kann den Winkel der Bindung nicht ändern! Fehlercode = ",GetLastError());
        return(false);
    };
//--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Ändert Text der Schaltfläche |
//+-----+
bool ButtonTextChange(const long   chart_ID=0,           // ID des Charts
                      const string name="Button",        // Name der Schaltfläche
                      const string text="Text")          // Text
{
//--- Setzen den Wert des Fehlers zurück

```

```

    ResetLastError();
//--- ändern wir den Text des Objektes
    if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
    {
        Print(__FUNCTION__,
              ": Konnte nicht den Text ändern! Fehlercode = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Löscht die Schaltfläche |
//+-----+
bool ButtonDelete(const long   chart_ID=0,    // ID des Charts
                  const string name="Button") // Name der Schaltfläche
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- die Schaltfläche löschen
    if(!ObjectDelete(chart_ID,name))
    {
        Print(__FUNCTION__,
              ": Könnte nicht die Schaltfläche löschen! Fehlercode = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//--- Größe des Chartfensters
    long x_distance;
    long y_distance;
//--- Größe des Chartfensters definieren
    if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
    {
        Print("Konnte nicht Breite des Charts bekommen! Fehlercode = ",GetLastError());
        return;
    }
    if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
    {
        Print("Konnte nicht Höhe des Charts bekommen! Fehlercode = ",GetLastError());
        return;
    }
//--- Schritt für Änderung der Schaltflächengröße definieren

```

```

int x_step=(int)x_distance/32;
int y_step=(int)y_distance/32;
//--- Schaltflächenkoordinaten und ihre Größe setzen
int x=(int)x_distance/32;
int y=(int)y_distance/32;
int x_size=(int)x_distance*15/16;
int y_size=(int)y_distance*15/16;
//--- eine Schaltfläche erstellen
if(!ButtonCreate(0,InpName,0,x,y,x_size,y_size,InpCorner,"Press",InpFont,InpFontSize,
    InpColor,InpBackColor,InpBorderColor,InpState,InpBack,InpSelection,InpHidden,Inp
    {
        return;
    }
; }
//--- den Chart neu zeichnen
ChartRedraw();
//--- Verkleinern die Schaltfläche im Zyklus
int i=0;
while(i<13)
{
    //--- Halb-Sekunden-Verzögerung
    Sleep(500);
    //--- Die Schaltfläche gedrückt setzen
    ObjectSetInteger(0,InpName,OBJPROP_STATE,true);
    //--- Die Grafik neu zeichnen und 0,2 Sekunden warten
    ChartRedraw();
    Sleep(200);
    //--- die Position und Größe der Schaltfläche neu definieren
    x+=x_step;
    y+=y_step;
    x_size-=x_step*2;
    y_size-=y_step*2;
    //--- Schaltfläche verkleinern
    ButtonMove(0,InpName,x,y);
    ButtonChangeSize(0,InpName,x_size,y_size);
    //--- Schaltfläche im ungedrückt Zustand setzen
    ObjectSetInteger(0,InpName,OBJPROP_STATE,false);
    //--- den Chart neu zeichnen
    ChartRedraw();
    //--- Überprüfen die Fakten von Zwangsabschaltung der Skript
    if(IsStopped())
        return;
    //--- Zykluszähler erhöhen
    i++;
}
//--- Halb-Sekunden-Verzögerung
Sleep(500);
//--- die Schaltfläche löschen
ButtonDelete(0,InpName);
ChartRedraw();

```



```
//--- 1 Sekunde warten  
    Sleep(1000);  
//---  
; }
```

## OBJ\_CHART

Objekt "Chart".



### Hinweis

Koordinaten des Ankerpunkts werden in Pixeln angegeben. Sie können den Winkel des Bindungspunktes durch Aufzählung [ENUM\\_BASE\\_CORNER](#) auswählen.

Für das Objekt "Chart" können Sie ein Symbol, Ausmaß und Periode Zeit, als auch aktivieren / deaktivieren Sie den Anzeigemodus der Skalen Preise und Termine auszuwählen.

### Beispiel

Das folgende Skript erstellt und bewegt auf dem Chart Objekt "Chart". Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Das Skript erstellt das Objekt \"Chart\"."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="Chart";           // Objektname
input string      InpSymbol="EURUSD";        // Symbol
input ENUM_TIMEFRAMES InpPeriod=PERIOD_H1;    // Periode
input ENUM_BASE_CORNER InpCorner=CORNER_LEFT_UPPER; // Winkel des Charts um Objekt zu
input int         InpScale=2;                 // Skala
input bool        InpDateScale=true;          // Anzeige der Zeitskala
input bool        InpPriceScale=true;         // Anzeige der Preisskala
```

```

input color      InpColor=clrRed;           // Rahmenfarbe bei dem Wählen
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Linienstil bei dem Wählen
input int        InpPointWidth=1;          // Größe des Punktes für Bewegung
input bool       InpBack=false;            // Objekt im Hintergrund
input bool       InpSelection=true;        // Wählen um zu bewegen
input bool       InpHidden=true;           // Ausgeblendet in der Liste der C
input long       InpZOrder=0;              // Priorität auf Mausklick

//+-----+
//| Erstellt das Objekt "Chart" |
//+-----+

bool ObjectChartCreate(const long      chart_ID=0,           // ID des Cha
                      const string    name="Chart",         // Objektname
                      const int       sub_window=0,         // Nummer des
                      const string    symbol="EURUSD",       // Symbol
                      const ENUM_TIMEFRAMES period=PERIOD_H1, // Periode
                      const int       x=0,                 // X-Koordinat
                      const int       y=0,                 // Y-Koordinat
                      const int       width=300,            // Breite
                      const int       height=200,           // Höhe
                      const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER, // Winkel des
                      const int       scale=2,             // Skala
                      const bool      date_scale=true,     // Anzeige de
                      const bool      price_scale=true,    // Anzeige de
                      const color      clr=clrRed,          // Rahmenfark
                      const ENUM_LINE_STYLE style=STYLE_SOLID, // Linienstil
                      const int       point_width=1,       // Größe des
                      const bool      back=false,          // Im Hintergr
                      const bool      selection=false,      // Markieren
                      const bool      hidden=true,          // Ausgeblenc
                      const long       z_order=0)           // Priorität

{
//--- Setzen den Wert des Fehlers zurück
ResetLastError();
//--- Erstellen wir das Objekt "Chart"
if(!ObjectCreate(chart_ID,name,OBJ_CHART,sub_window,0,0))
{
Print(__FUNCTION__,
      ": Objekt \"Chart\" konnte nicht erstellt werden! Fehlercode = ",GetLastEr
return(false);
};
//--- Objektkoordinaten angeben
ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- Objektgröße setzen
ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- wählen die Ecke des Charts, relativ zu der die Punktkoordinaten eingegeben werde
ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- den Symbol setzen

```

```

    ObjectSetString(chart_ID,name,OBJPROP_SYMBOL,symbol);
//--- den Zeitraum setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_PERIOD,period);
//--- die Skala setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_CHART_SCALE,scale);
//--- die Zeitskala anzeigen (true) oder verbergen (false)
    ObjectSetInteger(chart_ID,name,OBJPROP_DATE_SCALE,date_scale);
//--- die Preisskala anzeigen (true) oder verbergen (false)
    ObjectSetInteger(chart_ID,name,OBJPROP_PRICE_SCALE,price_scale);
//--- Farbe des Rahmens bei der Markierung des Objektes
    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- Linienstil des Rahmens bei der Markierung des Objektes
    ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- die Größe des Punktes, den benutzt werden kann, um ein Objekt zu bewegen
    ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,point_width);
//--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- Aktivieren (true) oder deaktivieren (false) Mausbewegung Modus
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der
    ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- setzen die Priorität für eine Mausklick-Ereignisse auf dem Chart
    ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
    return(true);
; }

//+-----+
//| Setzt den Symbol und den Zeitraum des Objekts "Chart" |
//+-----+
bool ObjectChartSetSymbolAndPeriod(const long      chart_ID=0,      // ID des C
                                   const string     name="Chart",    // Objektna
                                   const string     symbol="EURUSD",  // Symbol
                                   const ENUM_TIMEFRAMES period=PERIOD_H1) // Zeitraum
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Jetzt setzen wir den Symbol und den Zeitraum des Objekts "Chart"
    if(!ObjectSetString(chart_ID,name,OBJPROP_SYMBOL,symbol))
    {
        Print(__FUNCTION__,
              ": Konnte nicht den Symbol für Objekt \"Chart\" erstellen! Fehlercode = ",
              GetLastError());
        return(false);
    }
; }
    if(!ObjectSetInteger(chart_ID,name,OBJPROP_PERIOD,period))
    {
        Print(__FUNCTION__,
              ": Konnte nicht den Zeitraum für Objekt \"Chart\" erstellen! Fehlercode = ",
              GetLastError());
        return(false);
    }
; }

```

```

    ;}
//--- die erfolgreiche Umsetzung
    return(true);
    ;}
//+-----+
//| Bewegt das Objekt "Chart" |
//+-----+
bool ObjectChartMove(const long   chart_ID=0,    // ID des Charts (nicht des Objektes)
                    const string name="Chart",  // Objektname
                    const int    x=0,          // X-Koordinate
                    const int    y=0)         // Y-Koordinate
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Objekt bewegen
    if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
    {
        Print(__FUNCTION__,
              ": Kann nicht X-Koordinate des Objektes \"Chart\" bewegen! Fehlercode = ",
              GetLastError());
        return(false);
    }
    ;}
    if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
    {
        Print(__FUNCTION__,
              ": Kann nicht Y-Koordinate des Objektes \"Chart\" bewegen! Fehlercode = ",
              GetLastError());
        return(false);
    }
    ;}
//--- die erfolgreiche Umsetzung
    return(true);
    ;}
//+-----+
//| Ändert die Größe des Objektes "Chart" |
//+-----+
bool ObjectChartChangeSize(const long   chart_ID=0,    // ID des Charts (nicht des Objektes)
                          const string name="Chart",  // Objektname
                          const int    width=300,     // Breite
                          const int    height=200)    // Höhe
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Objektgröße ändern
    if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
    {
        Print(__FUNCTION__,
              ": Konnte nicht die Breite des Objekts \"Chart\" ändern! Fehlercode = ",
              GetLastError());
        return(false);
    }
    ;}
    if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
    {

```

```

        Print(__FUNCTION__,
              ": Konnte nicht die Höhe des Objekts \"Chart\" ändern! Fehlercode = ", GetLastError());
        return(false);
    };
}

//--- die erfolgreiche Umsetzung
return(true);
};

//+-----+
//| Gibt ID des Objektes "Chart" zurück          |
//+-----+
long ObjectChartGetID(const long   chart_ID=0,    // ID des Charts (nicht des Objekts)
                     const string name="Chart") // Objektname
{
    //--- eine Variable vorbereiten um ID des Objekts "Chart" zu erhalten
    long id=-1;
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- ID erhalten
    if(!ObjectGetInteger(chart_ID,name,OBJPROP_CHART_ID,0,id))
    {
        Print(__FUNCTION__,
              ": Konnte nicht ID des Objekts \"Chart\" erhalten! Fehlercode = ", GetLastError());
    };
    //--- Ergebnis zurückgeben
    return(id);
};

//+-----+
//| Löscht das Objekt "Chart"                      |
//+-----+
bool ObjectChartDelete(const long   chart_ID=0,    // ID des Charts (nicht des Objektes)
                      const string name="Chart") // Objektname
{
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Objekt löschen
    if(!ObjectDelete(chart_ID,name))
    {
        Print(__FUNCTION__,
              ": Objekt \"Chart\" konnte nicht gelöscht werden! Fehlercode = ", GetLastError());
        return(false);
    };
    //--- die erfolgreiche Umsetzung
    return(true);
};

//+-----+
//| Script program start function                    |
//+-----+
void OnStart()
{

```

```

//--- Die Anzahl der Symbolen in "Market Watch" erhalten
int symbols=SymbolsTotal(true);
//--- Bestimmen wir, ob es ein Symbol mit dem angegebenen Namen in der Liste der Symbole gibt
bool exist=false;
for(int i=0;i<symbols;i++)
    if(InpSymbol==SymbolName(i,true))
    {
        exist=true;
        break;
    };
if(!exist)
{
    Print("Fehler! Dieses Symbol ",InpSymbol," ist nicht in \"Market Watch\" verfügbar");
    return;
};
//--- Prüfung der Richtigkeit der Eingabeparameter
if(InpScale<0 || InpScale>5)
{
    Print("Fehler! Ungültige Eingabeparameter! ");
    return;
};

//--- Größe des Chartfensters
long x_distance;
long y_distance;
//--- Größe des Chartfensters definieren
if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
{
    Print("Konnte nicht Breite des Charts bekommen! Fehlercode = ",GetLastError());
    return;
};
if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
{
    Print("Konnte nicht Höhe des Charts bekommen! Fehlercode = ",GetLastError());
    return;
};
//--- Objektkoordinaten und seine Größe setzen
int x=(int)x_distance/16;
int y=(int)y_distance/16;
int x_size=(int)x_distance*7/16;
int y_size=(int)y_distance*7/16;
//--- Erstellen wir das Objekt "Chart"
if(!ObjectChartCreate(0,InpName,0,InpSymbol,InpPeriod,x,y,x_size,y_size,InpCorner,1,
    InpPriceScale,InpColor,InpStyle,InpPointWidth,InpBack,InpSelection,InpHidden,InpAlert))
{
    return;
};
//--- den Chart neu zeichnen und 1 Sekunde warten
ChartRedraw();

```

```

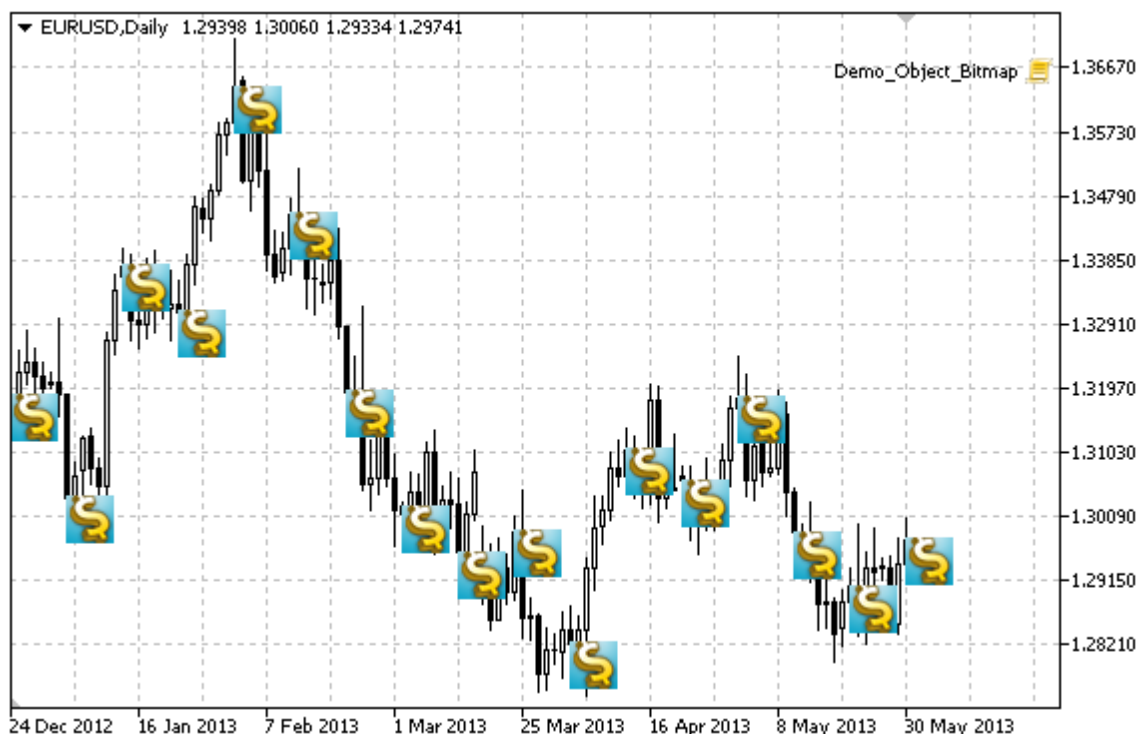
    Sleep(1000);
//--- Dehnen wir das Objekt "Chart"
    int steps=(int)MathMin(x_distance*7/16,y_distance*7/16);
    for(int i=0;i<steps;i++)
    {
        //--- Größe ändern
        x_size+=1;
        y_size+=1;
        if(!ObjectChartChangeSize(0,InpName,x_size,y_size))
            return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
        if(IsStopped())
            return;
        //--- Die Grafik neu zeichnen und 0,01 Sekunden warten
        ChartRedraw();
        Sleep(10);
    };
//--- Halb-Sekunden-Verzögerung
    Sleep(500);
//--- Zeitrahmen des Charts ändern
    if(!ObjectChartSetSymbolAndPeriod(0,InpName,InpSymbol,PERIOD_M1))
        return;
    ChartRedraw();
//--- Verzögerung von 3 Sekunden
    Sleep(3000);
//--- das Objekt löschen
    ObjectChartDelete(0,InpName);
    ChartRedraw();
//--- 1 Sekunde warten
    Sleep(1000);
//---
};

```



## OBJ\_BITMAP

Objekt "Bild".



### Hinweis

Für Objekt "Bild" können Sie [Scope](#) der Zeichnung.

### Beispiel

Das folgende Skript erstellt auf dem Chart einige Bilder. Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Das Skript erstellt ein Bild im Chartfenster."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpFile="\\Images\\dollar.bmp"; // Dateiname mit dem Bild
input int         InpWidth=24;                   // X-Koordinate des Rahmens
input int         InpHeight=24;                  // Y-Koordinate des Rahmens
input int         InpXOffset=4;                  // X-Offset des Rahmens
input int         InpYOffset=4;                  // Y-Offset des Rahmens
input color       InpColor=clrRed;               // Farbe des Rahmens bei der Zuteilung
input ENUM_LINE_STYLE InpStyle=STYLE_SOLID;      // Linienstil bei der Zuteilung
input int         InpPointWidth=1;              // Größe des Punktes für Bewegung
input bool        InpBack=false;                // Objekt im Hintergrund
input bool        InpSelection=false;           // Wählen zu bewegen
```

```

input bool          InpHidden=true;           // Ausgeblendet in der Liste der
input long          InpZOrder=0;              // Priorität auf Mausclick
//+-----+
//| Erzeugt ein Bild im Chartfenster |
//+-----+
bool BitmapCreate(const long      chart_ID=0,      // ID des Charts
                  const string    name="Bitmap",   // Bildname
                  const int       sub_window=0,    // Nummer des Unterfensters
                  datetime         time=0,         // Zeit des Ankerpunktes
                  double           price=0,        // Preis des Ankerpunktes
                  const string     file="",        // Name der Bilddatei
                  const int        width=10,      // X-Koordinate des Gültigkeitsbereichs
                  const int        height=10,     // Y-Koordinate des Gültigkeitsbereichs
                  const int        x_offset=0,    // X-Verschiebung des Gültigkeitsbereichs
                  const int        y_offset=0,    // Y-Verschiebung des Gültigkeitsbereichs
                  const color      clr=clrRed,    // Rahmenfarbe bei dem Wählen
                  const ENUM_LINE_STYLE style=STYLE_SOLID, // Linienstil bei dem Wählen
                  const int        point_width=1, // Größe des Ankerpunktes
                  const bool       back=false,    // Im Hintergrund
                  const bool       selection=false, // Auswählen um zu bewegen
                  const bool       hidden=true,   // Ausgeblendet in der Liste
                  const long        z_order=0)    // Priorität auf Mausclick
{
//--- Die Koordinaten des Bezugspunkts angeben, wenn sie nicht gesetzt sind
    ChangeBitmapEmptyPoint(time,price);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Erstellen wir ein Bild
    if(!ObjectCreate(chart_ID,name,OBJ_BITMAP,sub_window,time,price))
    {
        Print(__FUNCTION__,
              ": Konnte nicht das Bild im Chartfenster erstellen! Fehlercode = ",GetLastError());
        return(false);
    }
//--- Pfad zur Datei mit dem Bild setzen
    if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,file))
    {
        Print(__FUNCTION__,
              ": Bild konnte nicht geladen werden! Fehlercode = ",GetLastError());
        return(false);
    }
//--- Bildrahmen eingebenö wenn die Werte von Breite oder Höhe
//--- größer als die Breite und Höhe des ursprünglichen Bildes sind, dann
//--- wird es nicht gezeichnet; wenn die Werte von Breite und Höhe sind kleiner als die
//--- dann sein Teil, der diesen Größen entspricht, gezeichnet wird
    ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
    ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- setzen den Teil des Bildes, der in den Rahmen angezeigt werden soll
//--- Standardmäßig ist es der linke obere Bereich des Bildes; durch den Wert

```

```

//--- kann es von diesem Bild bewegt werden
    ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset);
    ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset);
//--- Farbe des Rahmens bei der Markierung des Objektes
    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- Linienstil des Rahmens bei der Markierung des Objektes
    ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- die Größe des Punktes, den benutzt werden kann, um ein Objekt zu bewegen
    ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,point_width);
//--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- Aktivieren (true) oder deaktivieren (false) Mausbewegung Modus
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der
    ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- setzen die Priorität für eine Mausklick-Ereignisse auf dem Chart
    ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
    return(true);
; }

//+-----+
//| Setzt ein neues Bild für die Zeichnung |
//+-----+
bool BitmapSetImage(const long   chart_ID=0,    // ID des Charts
                   const string name="Bitmap", // Name der Zeichnung
                   const string file="")       // Pfad zur Datei
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Pfad zur Datei mit dem Bild setzen
    if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,file))
    {
        Print(__FUNCTION__,
              ": Bild konnte nicht geladen werden! Fehlercode = ",GetLastError());
        return(false);
    }
; }
//--- die erfolgreiche Umsetzung
    return(true);
; }

//+-----+
//| Bewegt ein Bild im Chartfenster |
//+-----+
bool BitmapMove(const long   chart_ID=0,    // ID des Charts
               const string name="Bitmap", // Bildname
               datetime      time=0,       // Zeit des Ankerpunktes
               double        price=0)     // Preis des Ankerpunktes
{
//--- Wenn die Koordinaten des Punktes nicht angegeben sind, dann verschieben wir es

```

```

    if(!time)
        time=TimeCurrent();
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- bewegen den Bezugspunkt
    if(!ObjectMove(chart_ID,name,0,time,price))
    {
        Print(__FUNCTION__,
            ": der Bezugspunkt konnte nicht bewegt werden! Fehlercode = ",GetLastError());
        return(false);
    };
//--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Ändert die Größe von Gültigkeitsbereich (Bildgröße) |
//+-----+
bool BitmapChangeSize(const long   chart_ID=0,    // ID des Charts
                      const string name="Bitmap", // Bildname
                      const int    width=0,       // Breite des Bildes
                      const int    height=0)      // Höhe des Bildes
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Bildgröße ändern
    if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
    {
        Print(__FUNCTION__,
            ": Konnte nicht die Breite des Bildes ändern! Fehlercode = ",GetLastError());
        return(false);
    };
    if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
    {
        Print(__FUNCTION__,
            ": Konnte nicht die Höhe des Bildes ändern! Fehlercode = ",GetLastError());
        return(false);
    };
//--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Ändert die Koordinate der oberen linken Ecke von Bildrahmen |
//+-----+
bool BitmapMoveVisibleArea(const long   chart_ID=0,    // ID des Charts
                           const string name="Bitmap", // Bildname
                           const int    x_offset=0,    // X-Koordinate des Gültigkeitsbereichs
                           const int    y_offset=0)    // Y-Koordinate des Gültigkeitsbereichs

```

```

{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Koordinaten des Gültigkeitsbereichs des Bildes ändern
    if(!ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset))
    {
        Print(__FUNCTION__,
            ": X-Koordinate des Objektrahmens konnte nicht geändert werden! Fehlercode
        return(false);
    };
    if(!ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset))
    {
        Print(__FUNCTION__,
            ": Y-Koordinate des Objektrahmens konnte nicht geändert werden! Fehlercode
        return(false);
    };
//--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Löscht das Bild |
//+-----+
bool BitmapDelete(const long   chart_ID=0,    // ID des Charts
                  const string name="Bitmap") // Bildname
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- das Schild löschen
    if(!ObjectDelete(chart_ID,name))
    {
        Print(__FUNCTION__,
            ": Könnte nicht ein Bild löschen! Fehlercode = ",GetLastError());
        return(false);
    };
//--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Überprüft den Wert des Bezugspunkts für lehre Werte |
//| setzt Standardwerte |
//+-----+
void ChangeBitmapEmptyPoint(datetime &time,double &price)
{
//--- Wenn Punktzeit nicht angegeben ist, wird es auf dem aktuellen Bar sein
    if(!time)
        time=TimeCurrent();
//--- Wenn der Preis des Punktes nicht angegeben wird, dann wird es einen Wert von Bic
    if(!price)
        price=SymbolInfoDouble(Symbol(),SYMBOL_BID);

```

```

; }
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    datetime date[]; // Array speichert Daten von sichtbaren Bars
    double close[]; // Array speichert Close-Preise
    //-- Name der Bilddatei
    string file="\\Images\\dollar.bmp";
    //-- Die Anzahl der sichtbaren Bars im Chart-Fenster
    int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
    //-- Speicher reservieren
    ArrayResize(date,bars);
    ArrayResize(close,bars);
    //-- Füllen die Anordnung von Daten
    ResetLastError();
    if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
    {
        Print("Kann nicht den Wert der Zeit kopieren! Fehlercode = ",GetLastError());
        return;
    }
    ; }
    //-- Array mit Close-Preise Füllen
    if(CopyClose(Symbol(),Period(),0,bars,close)==-1)
    {
        Print("Kann nicht die Werte der Close-Preise kopieren! Fehlercode = ",GetLastError());
        return;
    }
    ; }
    //-- Definieren wir, wie oft das Bild anzuzeigen
    int scale=(int)ChartGetInteger(0,CHART_SCALE);
    //-- Schritt definieren
    int step=1;
    switch(scale)
    {
        case 0:
            step=27;
            break;
        case 1:
            step=14;
            break;
        case 2:
            step=7;
            break;
        case 3:
            step=4;
            break;
        case 4:
            step=2;
            break;
    }
}

```

```

    ;}
//--- Bilder für die Werte von High und Low der Baren erstellen (mit Zwischenräume)
for(int i=0;i<bars;i+=step)
{
    //--- Bilder erstellen
    if(!BitmapCreate(0,"Bitmap_"+(string)i,0,date[i],close[i],InpFile,InpWidth,InpHe
        InpYOffset,InpColor,InpStyle,InpPointWidth,InpBack,InpSelection,InpHidden,Inp
    {
        return;
    }
    ;}
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
if(IsStopped())
    return;
//--- den Chart neu zeichnen
ChartRedraw();
// Verzögerung 0.05 Sekunden
Sleep(50);
};
//--- Halb-Sekunden-Verzögerung
Sleep(500);
//--- Zeichen "Sell" löschen
for(int i=0;i<bars;i+=step)
{
    if(!BitmapDelete(0,"Bitmap_"+(string)i))
        return;
    if(!BitmapDelete(0,"Bitmap_"+(string)i))
        return;
    //--- den Chart neu zeichnen
    ChartRedraw();
    // Verzögerung 0.05 Sekunden
    Sleep(50);
};
//---
};

```

## OBJ\_BITMAP\_LABEL

Objekt "Bitmap Label".



### Hinweis

Die Position des Bezugspunktes relativ zum Zeichen kann aus der Aufzählung [ENUM\\_ANCHOR\\_POINT](#) gewählt werden. Koordinaten des Bezugspunktes werden in Pixel angegeben.

Sie können auch den Bezugswinkel von Bitmap Label aus der Aufzählung [ENUM\\_BASE\\_CORNER](#) wählen.

Für Objekt "Bitmap Label" können Sie [Scope](#) der Zeichnung.

### Beispiel

Das folgende Skript erstellt auf dem Chart einige Bilder. Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Das Skript erstellt Objekt \"Bitmap Label\"."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="BmpLabel";           // Name von Label
input string      InpFileOn="\\Images\\dollar.bmp"; // Der Dateiname für Modus (
input string      InpFileOff="\\Images\\euro.bmp"; // Der Dateiname für Modus (
input bool        InpState=false;               // Label gedrückt/ungedrückt
input ENUM_BASE_CORNER InpCorner=CORNER_LEFT_UPPER; // Der Winkel des Charts für
input ENUM_ANCHOR_POINT InpAnchor=ANCHOR_CENTER; // Bezugsmethode
```



```

input color      InpColor=clrRed;           // Farbe des Rahmens bei der Markierung
input ENUM_LINE_STYLE InpStyle=STYLE_SOLID; // Linienstil bei der Markierung
input int        InpPointWidth=1;          // Größe des Punktes zu bewegen
input bool       InpBack=false;            // Objekt im Hintergrund
input bool       InpSelection=false;       // Wählen zu bewegen
input bool       InpHidden=true;           // Ausgeblendet in der Liste
input long       InpZOrder=0;              // Priorität auf Mausclick

//+-----+
//| Erstellt das Objekt "Bitmap Label" |
//+-----+

bool BitmapLabelCreate(const long      chart_ID=0,           // ID des Charts
                       const string    name="BmpLabel",      // Name des Objekts
                       const int       sub_window=0,         // Nummer des Fensters
                       const int       x=0,                  // X-Koordinate
                       const int       y=0,                  // Y-Koordinate
                       const string     file_on="",           // Bild im Modus On
                       const string     file_off="",          // Bild im Modus Off
                       const int       width=0,               // X-Koordinate des Bildes
                       const int       height=0,             // Y-Koordinate des Bildes
                       const int       x_offset=10,           // X-Offset des Bildes
                       const int       y_offset=10,           // Y-Offset des Bildes
                       const bool       state=false,          // gedruckt/ausgedruckt
                       const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER, // Ecke des Objekts
                       const ENUM_ANCHOR_POINT anchor=ANCHOR_LEFT_UPPER, // Bindungsmechanismus
                       const color      clr=clrRed,           // Rahmenfarbe
                       const ENUM_LINE_STYLE style=STYLE_SOLID, // Linienstil
                       const int       point_width=1,         // Größe des Punktes
                       const bool       back=false,           // Im Hintergrund
                       const bool       selection=false,       // Markieren
                       const bool       hidden=true,           // Ausgeblendet
                       const long       z_order=0)             // Priorität

{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Bitmap Label erstellen
    if(!ObjectCreate(chart_ID,name,OBJ_BITMAP_LABEL,sub_window,0,0))
    {
        Print(__FUNCTION__,
              ": Objekt \"Bitmap Label\" konnte nicht erstellt werden! Fehlercode = ",GetLastError());
        return(false);
    };
//--- Wählen Bilder für Modi On und Off
    if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,0,file_on))
    {
        Print(__FUNCTION__,
              ": Bild für Modus On konnte nicht erstellt werden! Fehlercode = ",GetLastError());
        return(false);
    };
    if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,1,file_off))

```

```

    {
        Print(__FUNCTION__,
            ": Bild für Modus Off konnte nicht erstellt werden! Fehlercode = ", GetLastError());
        return(false);
    }
}

//--- Die Koordinaten des Schilds setzen
ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- Bildrahmen eingebenö wenn die Werte von Breite oder Höhe
//--- größer als die Breite und Höhe des ursprünglichen Bildes sind, dann
//--- wird es nicht gezeichnet; wenn die Werte von Breite und Höhe sind kleiner als d
//--- dann sein Teil, der diesen Größen entspricht, gezeichnet wird
ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- setzen den Teil des Bildes, der in den Rahmen angezeigt werden soll
//--- Standardmäßig ist es der linke obere Bereich des Bildes; durch den Wert
//--- kann es von diesem Bild bewegt werden
ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset);
ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset);
//--- Zustand von Label (gedrückt oder nicht)
ObjectSetInteger(chart_ID,name,OBJPROP_STATE,state);
//--- wählen die Ecke des Charts, relativ zu der die Punktkoordinaten eingegeben werde
ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- die Bindungsmethode setzen
ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- Farbe des Rahmens bei der Markierung des Objektes
ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- Linienstil des Rahmens bei der Markierung des Objektes
ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- die Größe des Punktes, den benutzt werden kann, um ein Objekt zu bewegen
ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,point_width);
//--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- Aktivieren (true) oder deaktivieren (false) Mausbewegung Modus
ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der
ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- setzen die Priorität für eine Mausklick-Ereignisse auf dem Chart
ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
return(true);
}

//+-----+
//| Ein neues Bild für "Bitmap Label" setzen      |
//+-----+
bool BitmapLabelSetImage(const long   chart_ID=0,      // ID des Charts
                        const string name="BmpLabel",  // Name des Objekts
                        const int    on_off=0,         // Modifier (On oder Off)

```

```

        const string file="")           // Pfad zur Datei
    {
//--- Setzen den Wert des Fehlers zurück
        ResetLastError();
//--- Pfad zur Datei mit dem Bild setzen
        if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,on_off,file))
        {
            Print(__FUNCTION__,
                ": Bild konnte nicht geladen werden! Fehlercode = ",GetLastError());
            return(false);
        };
//--- die erfolgreiche Umsetzung
        return(true);
    };
//+-----+
//| Verschiebt das Objekt "Bitmap Label" |
//+-----+
bool BitmapLabelMove(const long   chart_ID=0,      // ID des Charts
                    const string name="BmpLabel", // Name des Labels
                    const int    x=0,             // X-Koordinate
                    const int    y=0)             // Y-Koordinate
{
//--- Setzen den Wert des Fehlers zurück
        ResetLastError();
//--- Objekt bewegen
        if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
        {
            Print(__FUNCTION__,
                ": X-Koordinate des Objekts konnte nicht bewegt werden! Fehlercode = ",GetLastError());
            return(false);
        };
        if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
        {
            Print(__FUNCTION__,
                ": Y-Koordinate des Objekts konnte nicht bewegt werden! Fehlercode = ",GetLastError());
            return(false);
        };
//--- die erfolgreiche Umsetzung
        return(true);
    };
//+-----+
//| Ändert Rahmen des Objekts (Objektgröße) |
//+-----+
bool BitmapLabelChangeSize(const long   chart_ID=0,      // ID des Charts
                          const string name="BmpLabel", // Name des Labels
                          const int    width=0,         // Breite von Label
                          const int    height=0)         // Höhe von Label
{
//--- Setzen den Wert des Fehlers zurück

```

```

ResetLastError();
//--- Objektgröße ändern
if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
{
    Print(__FUNCTION__,
          ": Breite des Objekts konnte nicht geändert werden! Fehlercode = ",GetLastError());
    return(false);
};
if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
{
    Print(__FUNCTION__,
          ": Höhe des Objekts konnte nicht geändert werden! Fehlercode = ",GetLastError());
    return(false);
};
//--- die erfolgreiche Umsetzung
return(true);
};
//+-----+
//| Ändert die Koordinate der oberen linken Ecke von Bildrahmen |
//+-----+
bool BitmapLabelMoveVisibleArea(const long   chart_ID=0,      // ID des Charts
                                const string name="BmpLabel",  // Name des Objekts
                                const int    x_offset=0,       // X-Koordinate des Rahmens
                                const int    y_offset=0)        // Y-Koordinate des Rahmens
{
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Koordinaten des Objektrahmens ändern
    if(!ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset))
    {
        Print(__FUNCTION__,
              ": X-Koordinate des Objektrahmens konnte nicht geändert werden! Fehlercode = ",GetLastError());
        return(false);
    };
    if(!ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset))
    {
        Print(__FUNCTION__,
              ": Y-Koordinate des Objektrahmens konnte nicht geändert werden! Fehlercode = ",GetLastError());
        return(false);
    };
    //--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Löscht das Objekt "Bitmap Label" |
//+-----+
bool BitmapLabelDelete(const long   chart_ID=0,      // ID des Charts
                       const string name="BmpLabel") // Name des Objekts
{

```

```

//--- Setzen den Wert des Fehlers zurück
ResetLastError();
//--- das Schild löschen
if(!ObjectDelete(chart_ID,name))
{
    Print(__FUNCTION__,
        ": Objekt \"Bitmap Label\" konnte nicht gelöscht werden! Fehlercode = ",GetLastError());
    return(false);
};
//--- die erfolgreiche Umsetzung
return(true);
};
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    //--- Größe des Chartfensters
    long x_distance;
    long y_distance;
    //--- Größe des Chartfensters definieren
    if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
    {
        Print("Konnte nicht Breite des Charts bekommen! Fehlercode = ",GetLastError());
        return;
    };
    if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
    {
        Print("Konnte nicht Höhe des Charts bekommen! Fehlercode = ",GetLastError());
        return;
    };
    //--- Die Koordinaten des Objekts definieren
    int x=(int)x_distance/2;
    int y=(int)y_distance/2;
    //--- Objektgröße und Koordinate des Rahmens setzen
    int width=32;
    int height=32;
    int x_offset=0;
    int y_offset=0;
    //--- Bitmap Label auf der Mitte des Fensters platzieren
    if(!BitmapLabelCreate(0,InpName,0,x,y,InpFileOn,InpFileOff,width,height,x_offset,y_offset,
        InpCorner,InpAnchor,InpColor,InpStyle,InpPointWidth,InpBack,InpSelection,InpHide))
    {
        return;
    };
    //--- den Chart neu zeichnen und 1 Sekunde warten
    ChartRedraw();
    Sleep(1000);
    //--- Größe des Objektrahmens im Loop ändern

```

```
for(int i=0;i<6;i++)
{
    //--- Größe des Objektrahmens ändern
    width--;
    height--;
    if(!BitmapLabelChangeSize(0,InpName,width,height))
        return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
    if(IsStopped())
        return;
    //--- den Chart neu zeichnen
    ChartRedraw();
    // Verzögerung 0.3 Sekunden
    Sleep(300);
};
//--- 1 Sekunde Verzögerung
Sleep(1000);
//--- Koordinaten des Objektrahmens im Loop ändern
for(int i=0;i<2;i++)
{
    //--- Koordinaten des Objektrahmens ändern
    x_offset++;
    y_offset++;
    if(!BitmapLabelMoveVisibleArea(0,InpName,x_offset,y_offset))
        return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
    if(IsStopped())
        return;
    //--- den Chart neu zeichnen
    ChartRedraw();
    // Verzögerung 0.3 Sekunden
    Sleep(300);
};
//--- 1 Sekunde Verzögerung
Sleep(1000);
//--- das Schild löschen
BitmapLabelDelete(0,InpName);
ChartRedraw();
//--- 1 Sekunde Verzögerung
Sleep(1000);
//---
};
```

## OBJ\_EDIT

Object "Eingabefeld".



### Hinweis

Koordinaten des Ankerpunktes werden in Pixeln angegeben. Sie können den Winkel des Ankerpunktes des Objekts "Eingabefeld" durch Aufzählung [ENUM\\_BASE\\_CORNER](#) auswählen.

Sie können auch eine der Ausrichtung von Text innerhalb der "Eingabefelds" in Aufzählung [ENUM\\_ALIGN\\_MODE](#) wählen.

### Beispiel

Das folgende Skript erstellt und bewegt auf dem Chart Objekt "Eingabefeld". Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Das Skript erstellt das Objekt \"Eingabefeld\"."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="Edit";           // Objektname
input string      InpText="Text";           // Objekttext
input string      InpFont="Arial";          // Schrift
input int         InpFontSize=14;           // Schriftgröße
input ENUM_ALIGN_MODE InpAlign=ALIGN_CENTER; // Textausrichtung
input bool        InpReadOnly=false;        // Bearbeitung
input ENUM_BASE_CORNER InpCorner=CORNER_LEFT_UPPER; // Winkel des Charts zu binden
input color       InpColor=clrBlack;        // Textfarbe
```

```

input color      InpBackColor=clrWhite;      // Hintergrundfarbe
input color      InpBorderColor=clrBlack;    // Rahmenfarbe
input bool       InpBack=false;              // Objekt im Hintergrund
input bool       InpSelection=false;         // Wählen zu bewegen
input bool       InpHidden=true;             // Ausgeblendet in der Liste der C
input long       InpZOrder=0;                // Priorität auf Mausklick

//+-----+
//| Erstellt das Objekt "Eingabefeld"          |
//+-----+

bool EditCreate(const long      chart_ID=0,      // ID des Charts
                const string    name="Edit",    // Objektname
                const int       sub_window=0,    // Nummer des Unterfe
                const int       x=0,             // X-Koordinate
                const int       y=0,             // Y-Koordinate
                const int       width=50,        // Breite
                const int       height=18,       // Höhe
                const string     text="Text",    // Text
                const string     font="Arial",   // Schrift
                const int       font_size=10,    // Schriftgröße
                const ENUM_ALIGN_MODE align=ALIGN_CENTER, // Textausrichtung
                const bool       read_only=false, // Bearbeitung
                const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER, // Winkel des Charts
                const color      clr=clrBlack,   // Textfarbe
                const color      back_clr=clrWhite, // Hintergrundfarbe
                const color      border_clr=clrNONE, // Rahmenfarbe
                const bool       back=false,     // Im Hintergrund
                const bool       selection=false, // Wählen um zu bewege
                const bool       hidden=true,    // Ausgeblendet in de
                const long       z_order=0)      // Priorität auf Maus

{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Eingabefeld erstellen
    if(!ObjectCreate(chart_ID,name,OBJ_EDIT,sub_window,0,0))
    {
        Print(__FUNCTION__,
              ": Objekt \"Eingabefeld\" konnte nicht erstellt werden! Fehlercode = ",Get
        return(false);
    }
//--- Objektkoordinaten angeben
    ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
    ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- Objektgröße setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
    ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- den Text setzen
    ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- Textschrift setzen
    ObjectSetString(chart_ID,name,OBJPROP_FONT,font);

```



```

//--- Schriftgröße setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
//--- Modus von Textausrichtung im Objekt setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_ALIGN,align);
//--- aktivieren (true) oder deaktivieren (false) den schreibgeschützten Modus
    ObjectSetInteger(chart_ID,name,OBJPROP_READONLY,read_only);
//--- setzen die Ecke des Diagramms, in Bezug auf die die Koordinaten des Objekts best
    ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- Textfarbe setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- Hintergrundfarbe setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_BGCOLOR,back_clr);
//--- Rahmenfarbe setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_COLOR,border_clr);
//--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- Aktivieren (true) oder deaktivieren (false) Mausbewegung Modus
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der
    ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- setzen die Priorität für eine Mausklick-Ereignisse auf dem Chart
    ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
    return(true);
; }

//+-----+
//| Bewegt das Objekt "Eingabefeld" |
//+-----+

bool EditMove(const long   chart_ID=0, // ID des Charts
              const string name="Edit", // Objektname
              const int    x=0,        // X-Koordinate
              const int    y=0)        // Y-Koordinate
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Objekt bewegen
    if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
    {
        Print(__FUNCTION__,
              ": X-Koordinate des Objekts konnte nicht bewegt werden! Fehlercode = ",GetLastError());
        return(false);
    }
; }
    if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
    {
        Print(__FUNCTION__,
              ": Y-Koordinate des Objekts konnte nicht bewegt werden! Fehlercode = ",GetLastError());
        return(false);
    }
; }

```

```

//--- die erfolgreiche Umsetzung
    return(true);
; }
//+-----+
//| Ändert die Größe des Objektes "Eingabefeld" |
//+-----+
bool EditChangeSize(const long   chart_ID=0, // ID des Charts
                   const string name="Edit", // Objektname
                   const int    width=0,    // Breite
                   const int    height=0)   // Höhe
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Objektgröße ändern
    if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
    {
        Print(__FUNCTION__,
              ": Breite des Objekts konnte nicht geändert werden! Fehlercode = ",GetLastError());
        return(false);
    ; }
    if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
    {
        Print(__FUNCTION__,
              ": Höhe des Objekts konnte nicht geändert werden! Fehlercode = ",GetLastError());
        return(false);
    ; }
//--- die erfolgreiche Umsetzung
    return(true);
; }
//+-----+
//| Ändert den Text des Objektes "Eingabefeld" |
//+-----+
bool EditTextChange(const long   chart_ID=0, // ID des Charts
                   const string name="Edit", // Objektname
                   const string text="Text") // Text
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- ändern wir den Text des Objektes
    if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
    {
        Print(__FUNCTION__,
              ": Konnte nicht den Text ändern! Fehlercode = ",GetLastError());
        return(false);
    ; }
//--- die erfolgreiche Umsetzung
    return(true);
; }
//+-----+

```

```

//| Gibt den Text des Objektes "Eingabefeld" zurück |
//+-----+
bool EditTextGet(string      &text,          // Text
                 const long  chart_ID=0,    // ID des Charts
                 const string name="Edit") // Objektname
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- erhalten wir den Text des Objektes
    if(!ObjectGetString(chart_ID,name,OBJPROP_TEXT,0,text))
    {
        Print(__FUNCTION__,
              ": Konnte nicht den Text erhalten! Fehlercode = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Löscht das Objekt "Eingabefeld" |
//+-----+
bool EditDelete(const long  chart_ID=0,    // ID des Charts
                const string name="Edit") // Objektname
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- das Schild löschen
    if(!ObjectDelete(chart_ID,name))
    {
        Print(__FUNCTION__,
              ": Objekt \"Eingabefeld\" konnte nicht gelöscht werden! Fehlercode = ",GetLastError());
        return(false);
    }
//--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//--- Größe des Chartfensters
    long x_distance;
    long y_distance;
//--- Größe des Chartfensters definieren
    if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
    {
        Print("Konnte nicht Breite des Charts bekommen! Fehlercode = ",GetLastError());
        return;
    }

```

```

    };
    if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
    {
        Print("Konnte nicht Höhe des Charts bekommen! Fehlercode = ",GetLastError());
        return;
    };
    //--- Schritt für Änderung der Eingabefeldgröße definieren
    int x_step=(int)x_distance/64;
    //--- Eingabefeldkoordinaten und seine Größe setzen
    int x=(int)x_distance/8;
    int y=(int)y_distance/2;
    int x_size=(int)x_distance/8;
    int y_size=InpFontSize*2;
    //--- den Text in einer lokalen Variable speichern
    string text=InpText;
    //--- Eingabefeld erstellen
    if(!EditCreate(0,InpName,0,x,y,x_size,y_size,InpText,InpFont,InpFontSize,InpAlign,1,
        InpCorner,InpColor,InpBackColor,InpBorderColor,InpBack,InpSelection,InpHidden,Inp
    {
        return;
    };
    //--- den Chart neu zeichnen und 1 Sekunde warten
    ChartRedraw();
    Sleep(1000);
    //--- Eingabefeld strecken
    while(x_size-x<x_distance*5/8)
    {
        //--- Breite der Eingabefeld erhöhen
        x_size+=x_step;
        if(!EditChangeSize(0,InpName,x_size,y_size))
            return;
    }
    //--- Überprüfen die Fakten von Zwangsabschaltung der Skript
    if(IsStopped())
        return;
    //--- Die Grafik neu zeichnen und 0,05 Sekunden warten
    ChartRedraw();
    Sleep(50);
    };
    //--- Halb-Sekunden-Verzögerung
    Sleep(500);
    //--- Text ändern
    for(int i=0;i<20;i++)
    {
        //--- "+" am Anfang und am Ende hinzufügen
        text="+"+text+" ";
        if(!EditTextChange(0,InpName,text))
            return;
    }
    //--- Überprüfen die Fakten von Zwangsabschaltung der Skript
    if(IsStopped())

```

```
        return;

        //--- Die Grafik neu zeichnen und 0,1 Sekunden warten
        ChartRedraw();
        Sleep(100);
    };
//--- Halb-Sekunden-Verzögerung
    Sleep(500);
//--- Eingabefeld löschen
    EditDelete(0, InpName);
    ChartRedraw();
//--- 1 Sekunde warten
    Sleep(1000);
//---
};
```

## OBJ\_EVENT

Objekt "Ereignis".



### Hinweis

Wenn Sie den Mauszeiger auf das Ereignis schweben, erscheint sein Text.

### Beispiel

Das folgende Skript erstellt und bewegt auf dem Chart Objekt "Ereignis". Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Das Skript erstellt ein graphisches Objekt \"Ereignis\"."
#property description "Datum des Ankerpunkts wird als Prozentsatz"
#property description "des Breite des Chartfensters angegeben."
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="Event";      // Name des Ereignisses
input int         InpDate=25;           // Datum des Ereignisses in %
input string      InpText="Text";       // Text des Ereignisses
input color       InpColor=clrRed;      // Farbe des Ereignisses
input int         InpWidth=1;           // Größe des Punktes für Bewegung
input bool        InpBack=false;        // Ereignis Im Hintergrund
input bool        InpSelection=false;   // Wählen um zu bewegen
input bool        InpHidden=true;       // Ausgeblendet in der Objektliste
```

```

input long          InpZOrder=0;          // Priorität auf Mausklick
//+-----+
//| Erstellt das Objekt "Ereignis" auf dem Chart |
//+-----+
bool EventCreate(const long          chart_ID=0,          // ID des Charts
                 const string        name="Event",        // Name des Ereignisses
                 const int           sub_window=0,        // Nummer des Unterfensters
                 const string        text="Text",         // Text des Ereignisses
                 datetime            time=0,              // Zeit
                 const color         clr=clrRed,          // Farbe
                 const int           width=1,             // Punktdicke bei Wählen
                 const bool          back=false,          // Im Hintergrund
                 const bool          selection=false,     // Wählen um zu bewegen
                 const bool          hidden=true,         // Ausgeblendet in der Objektliste
                 const long          z_order=0)           // Priorität auf Mausklick
{
//--- wenn die Zeit nicht angegeben ist, wird Objekt auf dem letzten Balken erstellt v
    if(!time)
        time=TimeCurrent();
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- Erstellen wir das Objekt "Ereignis"
    if(!ObjectCreate(chart_ID,name,OBJ_EVENT,sub_window,time,0))
    {
        Print(__FUNCTION__,
              ": Objekt \"Ereignis\" konnte nicht erstellt werden! Fehlercode = ",GetLastError());
        return(false);
    };
//--- den Text des Ereignisses setzen
    ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- Farbe setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- Dicke des Ankerpunktes, wenn ein Objekt ausgewählt ist, setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- aktivieren (true) oder deaktivieren (false) Bewegung des Ereignisses mit dem Mauszeiger
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der Objektliste
    ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- setzen die Priorität für eine Mausklick-Ereignisse auf dem Chart
    ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Ändert den Text des Objektes "Ereignis" |
//+-----+

```

```

bool EventTextChange(const long   chart_ID=0,    // ID des Charts
                    const string name="Event",  // Name des Ereignisses
                    const string text="Text")    // Text
{
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- ändern wir den Text des Objektes
    if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
    {
        Print(__FUNCTION__,
              ": Konnte nicht den Text ändern! Fehlercode = ",GetLastError());
        return(false);
    }
    //--- die erfolgreiche Umsetzung
    return(true);
}

//+-----+
//| Bewegung des Objekts "Ereignis" |
//+-----+
bool EventMove(const long   chart_ID=0,    // ID des Charts
              const string name="Event",  // Name des Ereignisses
              datetime      time=0)        // Zeit
{
    //--- wenn die Zeit nicht angegeben ist, bewegen wir das Ereignis auf den letzten Ball
    if(!time)
        time=TimeCurrent();
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Objekt bewegen
    if(!ObjectMove(chart_ID,name,0,time,0))
    {
        Print(__FUNCTION__,
              ": Objekt \"Ereignis\" konnte nicht bewegt werden! Fehlercode = ",GetLastError());
        return(false);
    }
    //--- die erfolgreiche Umsetzung
    return(true);
}

//+-----+
//| Löscht das Objekt "Ereignis" |
//+-----+
bool EventDelete(const long   chart_ID=0,    // ID des Charts
                const string name="Event")   // Name des Ereignisses
{
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- das Objekt löschen
    if(!ObjectDelete(chart_ID,name))
    {

```



```

        Print(__FUNCTION__,
              ": Objekt \"Ereignis\" konnte nicht gelöscht werden! Fehlercode = ",GetLastError());
        return(false);
    };
}

//--- die erfolgreiche Umsetzung
return(true);
};

//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    //--- Überprüfen die Richtigkeit der Eingabeparameter
    if(InpDate<0 || InpDate>100)
    {
        Print("Fehler! Ungültige Eingabeparameter! ");
        return;
    };
    //--- Die Anzahl der sichtbaren Bars im Chart-Fenster
    int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
    //--- Array speichert Datumswerte, die verwendet werden,
    //--- um die Koordinaten der Ankerpunkten des Objekts "Ereignis" zu setzen und zu bewegen
    datetime date[];
    //--- Speicher reservieren
    ArrayResize(date,bars);
    //--- Füllen die Anordnung von Daten
    ResetLastError();
    if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
    {
        Print("Kann nicht den Wert der Zeit kopieren! Fehlercode = ",GetLastError());
        return;
    };
    //--- Definieren die Punkte um das Objekt zu erstellen
    int d=InpDate*(bars-1)/100;
    //--- Erstellen wir das Objekt "Ereignis"
    if(!EventCreate(0,InpName,0,InpText,date[d],InpColor,InpWidth,
                  InpBack,InpSelection,InpHidden,InpZOrder))
    {
        return;
    };
    //--- den Chart neu zeichnen und 1 Sekunde warten
    ChartRedraw();
    Sleep(1000);
    //--- jetzt bewegen wir das Objekt
    //--- Schleifenzähler
    int h_steps=bars/2;
    //--- Objekt bewegen
    for(int i=0;i<h_steps;i++)
    {

```

```
//--- den nächsten Wert nehmen
if(d<bars-1)
    d+=1;
//--- den Punkt bewegen
if(!EventMove(0,InpName,date[d]))
    return;
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
if(IsStopped())
    return;
//--- den Chart neu zeichnen
ChartRedraw();
// Verzögerung 0.05 Sekunden
Sleep(50);
; }
//--- 1 Sekunde Verzögerung
Sleep(1000);
//--- den Kanal aus dem Chart löschen
EventDelete(0,InpName);
ChartRedraw();
//--- 1 Sekunde Verzögerung
Sleep(1000);
//---
; }
```

## OBJ\_RECTANGLE\_LABEL

Objekt "Rechteckiges Label."



### Hinweis

Koordinaten des Ankerpunkts werden in Pixeln angegeben. Sie können den Winkel des Anbindepunktes des rechteckigen Labels durch Aufzählung [ENUM\\_BASE\\_CORNER](#) wählen. Rahmentyp für ein rechteckiges Label kann durch [ENUM\\_BORDER\\_TYPE](#) gesetzt werden.

Es wird für Erstellung und Gestaltung der grafischen Benutzerschnittstelle verwendet.

### Beispiel

Das folgende Skript erstellt und bewegt auf dem Chart Objekt "Rechteckiges Label". Um die Eigenschaften eines grafischen Objekts zu erstellen und ändern, gibt es spezielle Funktionen, die Sie "wie es ist" für Ihre eigenen Programme verwenden können.

```
//--- Beschreibung
#property description "Das Skript erstellt ein graphisches Objekt \"Rechteckiges Label\"
//--- Zeigen den Fenster von Eingabeparametern in der Skript-Startup
#property script_show_inputs
//--- Eingangsparameter von Skript
input string      InpName="RectLabel";           // Label Name
input color       InpBackColor=clrSkyBlue;       // Hintergrundfarbe
input ENUM_BORDER_TYPE InpBorder=BORDER_FLAT;    // Rahmentyp
input ENUM_BASE_CORNER InpCorner=CORNER_LEFT_UPPER; // Winkel des Charts zu binden
input color       InpColor=clrDarkBlue;         // Die Farbe des flachen Randes (Flat)
input ENUM_LINE_STYLE InpStyle=STYLE_SOLID;      // Stil des flachen Randes (Flat)
input int         InpLineWidth=3;               // Dicke des flachen Randes (Flat)
```

```

input bool      InpBack=false;           // Objekt im Hintergrund
input bool      InpSelection=true;       // Wählen um zu bewegen
input bool      InpHidden=true;         // Ausgeblendet in der Liste der C
input long      InpZOrder=0;            // Priorität auf Mausclick
//+-----+
//| Erstellt ein rechteckiges Label |
//+-----+
bool RectLabelCreate(const long      chart_ID=0,           // ID des Charts
                    const string    name="RectLabel",     // Label Name
                    const int       sub_window=0,         // Nummer des Ur
                    const int       x=0,                 // X-Koordinate
                    const int       y=0,                 // Y-Koordinate
                    const int       width=50,            // Breite
                    const int       height=18,           // Höhe
                    const color     back_clr=C'236,233,216', // Hintergrundfa
                    const ENUM_BORDER_TYPE border=BORDER_SUNKEN, // Rahmentyp
                    const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER, // Winkel des Ch
                    const color     clr=clrRed,           // Die Farbe des
                    const ENUM_LINE_STYLE style=STYLE_SOLID, // Stil des flac
                    const int       line_width=1,        // Breite des fl
                    const bool      back=false,          // Im Hintergrun
                    const bool      selection=false,     // Wählen um zu
                    const bool      hidden=true,         // Ausgeblendet
                    const long      z_order=0)           // Priorität auf

{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- ein rechteckiges Label erstellen
    if(!ObjectCreate(chart_ID,name,OBJ_RECTANGLE_LABEL,sub_window,0,0))
    {
        Print(__FUNCTION__,
              ": rechteckiges Label konnte nicht erstellt werden! Fehlercode = ",GetLast
        return(false);
    }
; }
//--- Die Koordinaten des Schilds setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
    ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- Labelgröße setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
    ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- Hintergrundfarbe setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_BGCOLOR,back_clr);
//--- Rahmentyp setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_TYPE,border);
//--- wählen die Ecke des Charts, relativ zu der die Punktkoordinaten eingegeben werde
    ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- Die Farbe des flachen Randes (in Modus Flat) setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- Linienstil des flachen Rahmens setzen

```

```

    ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- Breite des flachen Rahmens setzen
    ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,line_width);
//--- Im Vordergrund (false) oder Hintergrund (true) anzeigen
    ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- Aktivieren (true) oder deaktivieren (false) Mausbewegung Modus
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
    ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- Verbergen (true) oder Anzeigen (false) den Namen des graphischen Objektes in der
    ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- setzen die Priorität für eine Mausklick-Ereignisse auf dem Chart
    ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Bewegt ein rechteckiges Label |
//+-----+
bool RectLabelMove(const long   chart_ID=0,      // ID des Charts
                  const string name="RectLabel", // Label Name
                  const int    x=0,             // X-Koordinate
                  const int    y=0)             // Y-Koordinate
{
//--- Setzen den Wert des Fehlers zurück
    ResetLastError();
//--- ein rechteckiges Label bewegen
    if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
    {
        Print(__FUNCTION__,
              ": Kann nicht X-Koordinate des Labels bewegen! Fehlercode = ",GetLastError());
        return(false);
    };
    if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
    {
        Print(__FUNCTION__,
              ": Kann nicht Y-Koordinate des Labels bewegen! Fehlercode = ",GetLastError());
        return(false);
    };
//--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Ändert die Größe eines rechteckigen Labels |
//+-----+
bool RectLabelChangeSize(const long   chart_ID=0,      // ID des Charts
                        const string name="RectLabel", // Name des Labels
                        const int    width=50,         // Breite des Labels
                        const int    height=18)        // Höhe des Labels
{

```

```

//--- Setzen den Wert des Fehlers zurück
ResetLastError();
//--- Größe des Labels ändern
if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
{
    Print(__FUNCTION__,
        ": Konnte nicht die Breite des Labels ändern! Fehlercode = ",GetLastError()
    return(false);
};
if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
{
    Print(__FUNCTION__,
        ": Konnte nicht die Höhe des Labels ändern! Fehlercode = ",GetLastError()
    return(false);
};
//--- die erfolgreiche Umsetzung
return(true);
};
//+-----+
//| Ändert den Rahmentyp des rechteckigen Labels |
//+-----+
bool RectLabelChangeBorderType(const long      chart_ID=0,          // ID des
                               const string     name="RectLabel",    // Name de
                               const ENUM_BORDER_TYPE border=BORDER_SUNKEN) // Rahment
{
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- Rahmentyp ändern
    if(!ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_TYPE,border))
    {
        Print(__FUNCTION__,
            ": Konnte nicht Rahmentyp ändern! Fehlercode = ",GetLastError());
        return(false);
    };
    //--- die erfolgreiche Umsetzung
    return(true);
};
//+-----+
//| Löscht ein rechteckiges Label |
//+-----+
bool RectLabelDelete(const long  chart_ID=0,          // ID des Charts
                     const string name="RectLabel") // Name des Labels
{
    //--- Setzen den Wert des Fehlers zurück
    ResetLastError();
    //--- das Schild löschen
    if(!ObjectDelete(chart_ID,name))
    {
        Print(__FUNCTION__,

```

```

        ": rechteckiges Label konnte nicht gelöscht werden! Fehlercode = ", GetLastError();
        return(false);
    };
}

//--- die erfolgreiche Umsetzung
return(true);
};

//+-----+
//| Script program start function |
//+-----+

void OnStart()
{
    //--- Größe des Chartfensters
    long x_distance;
    long y_distance;
    //--- Größe des Chartfensters definieren
    if(!ChartGetInteger(0, CHART_WIDTH_IN_PIXELS, 0, x_distance))
    {
        Print("Konnte nicht Breite des Charts bekommen! Fehlercode = ", GetLastError());
        return;
    };
    if(!ChartGetInteger(0, CHART_HEIGHT_IN_PIXELS, 0, y_distance))
    {
        Print("Konnte nicht Höhe des Charts bekommen! Fehlercode = ", GetLastError());
        return;
    };
    //--- Koordinaten eines rechteckigen Labels definieren
    int x=(int)x_distance/4;
    int y=(int)y_distance/4;
    //--- Labelgröße setzen
    int width=(int)x_distance/4;
    int height=(int)y_distance/4;
    //--- ein rechteckiges Label erstellen
    if(!RectLabelCreate(0, InpName, 0, x, y, width, height, InpBackColor, InpBorder, InpCorner,
        InpColor, InpStyle, InpLineWidth, InpBack, InpSelection, InpHidden, InpZOrder))
    {
        return;
    };
    //--- den Chart neu zeichnen und 1 Sekunde warten
    ChartRedraw();
    Sleep(1000);
    //--- Größe des rechteckigen Labels ändern
    int steps=(int)MathMin(x_distance/4, y_distance/4);
    for(int i=0; i<steps; i++)
    {
        //--- Größe ändern
        width+=1;
        height+=1;
        if(!RectLabelChangeSize(0, InpName, width, height))
            return;
    }
}

```

```
//--- Überprüfen die Fakten von Zwangsabschaltung der Skript
    if(IsStopped())
        return;
    //--- Die Grafik neu zeichnen und 0,01 Sekunden warten
    ChartRedraw();
    Sleep(10);
; }

//--- 1 Sekunde Verzögerung
Sleep(1000);

//--- Rahmentyp ändern
if(!RectLabelChangeBorderType(0, InpName, BORDER_RAISED))
    return;

//--- den Chart neu zeichnen und 1 Sekunde warten
ChartRedraw();
Sleep(1000);

//--- Rahmentyp ändern
if(!RectLabelChangeBorderType(0, InpName, BORDER_SUNKEN))
    return;

//--- den Chart neu zeichnen und 1 Sekunde warten
ChartRedraw();
Sleep(1000);

//--- das Schild löschen
RectLabelDelete(0, InpName);
ChartRedraw();

//--- 1 Sekunde warten
Sleep(1000);

//---
; }
```



## Objekteigenschaften

Jedes graphisches Objekt auf dem Preischart hat bestimmten Eigenschaftsvorrat. Einstellung und Erhalten von Werten der Objekteigenschaften erfolgt durch die entsprechenden [Funktionen für Arbeit mit graphischen Objekten](#). Für jeden [Objekttyp](#) gibt es Eigenschaftsvorrat, hier werden alle möglichen Werte aus der Aufzählungsfamilie `ENUM_OBJECT_PROPERTY` aufgezählt. Einige Eigenschaften müssen präzisiert werden, zB. Levelnummer für Objekt der Erweiterung `Fibonacci`. In diesen Fällen ist es notwendig, in Funktionen `ObjectSet...` und `ObjectGet...` den Wert des Parameters *modifier* anzugeben.

Die Funktionen, die die Eigenschaften von grafischen Objekten eingeben, und die Operationen der Erstellung `ObjectCreate()` und Bewegen `ObjectMove()` von Objekten auf dem Chart sind tatsächlich verwendet, um Befehle dem Chart zu senden. Der erfolgreiche Erfüllung dieser Funktionen wird der Befehl in die allgemeine Warteschlange der Chartereignisse platziert. Visuelle Veränderung der Eigenschaften von grafischen Objekten wird während der Verarbeitung der Ereigniswarteschlange von diesem Chart geändert.

Aus diesem Grund sollte man nicht erwarten, dass die graphischen Objekten sofort nach dem Aufruf dieser Funktionen visuell aktualisiert wird. Im Allgemeinen werden die graphischen Objekten durch das Terminal automatisch nach Änderung-Ereignisse aktualisiert - die Ankunft der neuen Quote, Änderung von Chartfenstergröße, etc.

Um graphischen Objekten zu aktualisieren, verwenden Sie [ChartRedraw\(\)](#).

für die Funktionen [ObjectSetInteger\(\)](#) und [ObjectGetInteger\(\)](#)

### ENUM\_OBJECT\_PROPERTY\_INTEGER

| Identifikator               | Beschreibung                                                                                                                                                                                                                                                                                                                                                                             | Typ der Eigenschaft             |
|-----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------|
| <code>OBJPROP_COLOR</code>  | Farbe                                                                                                                                                                                                                                                                                                                                                                                    | color                           |
| <code>OBJPROP_STYLE</code>  | Stil                                                                                                                                                                                                                                                                                                                                                                                     | <a href="#">ENUM_LINE_STYLE</a> |
| <code>OBJPROP_WIDTH</code>  | Weite von Linie                                                                                                                                                                                                                                                                                                                                                                          | int                             |
| <code>OBJPROP_BACK</code>   | Objekt auf dem Hintergrund                                                                                                                                                                                                                                                                                                                                                               | bool                            |
| <code>OBJPROP_ZORDER</code> | Priorität eines graphischen Objects für Empfang des Ereignisses einen Mausklick auf dem Chart ( <a href="#">CHARTEVENT_CLICK</a> ).<br>Standardmäßig ist der Wert bei Erstellen auf Null gesetzt, aber Sie können die Priorität erhöhen. Bei der Anwendung der Objekte auf jeder anderen, nur ein Objekt mit der höchsten Priorität bekommt das Ereignis <code>CHARTEVENT_CLICK</code> . | long                            |
| <code>OBJPROP_FILL</code>   | Füllen ein Objekt mit Farbe                                                                                                                                                                                                                                                                                                                                                              | bool                            |

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                                                            |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------|
|                    | (für OBJ_RECTANGLE, OBJ_TRIANGLE, OBJ_ELLIPSE, OBJ_CHANNEL, OBJ_STDDEVCHANNEL, OBJ_REGRESSION)                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                                                            |
| OBJPROP_HIDDEN     | Anzeige des Namens von graphischem Objekt in der Objektliste aus dem Terminal-Menü "Charts" - "Objekte" - "Objektliste". Wert true ermöglicht es, ein Objekt aus der Liste zu verstecken. Standardmäßig wird true für die Objekte, die Kalenderereignisse und Handelsgeschichte zeigen, und auch für die Objekte, die <a href="#">aus MQL5-Programmen erstellt werden</a> , angegeben. Um solche graphischen Objekte <a href="#">graphischen Objekte</a> zu sehen und Zugang zu ihren Eigenschaften zu haben, klicken Sie auf "Alle" im Fenster "Objektliste". | bool                                                       |
| OBJPROP_SELECTED   | Hervorheben des Objektes                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | bool                                                       |
| OBJPROP_READONLY   | Möglichkeit von Texteditieren im Objekt Edit                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | bool                                                       |
| OBJPROP_TYPE       | Objekttyp                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | <a href="#">ENUM_OBJECT</a> r/o                            |
| OBJPROP_TIME       | Koordinate der Zeit                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | datetime<br>Modifikator=Nummer des Bezugspunktes           |
| OBJPROP_SELECTABLE | Zugänglichkeit des Objekts                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | bool                                                       |
| OBJPROP_CREATETIME | Erzeugungszeit des Objekts                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | datetime r/o                                               |
| OBJPROP_LEVELS     | Levelszahl                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | int                                                        |
| OBJPROP_LEVELCOLOR | Farbe von Linie-Level                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | color<br>Modifikator=Levelnummer                           |
| OBJPROP_LEVELSTYLE | Stil der Linie-Level                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | <a href="#">ENUM_LINE_STYLE</a><br>Modifikator=Levelnummer |
| OBJPROP_LEVELWIDTH | Dicke der Linie-Level                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | int<br>Modifikator=Levelnummer                             |
| OBJPROP_ALIGN      | Horizontale Textausrichtung in der "Edit"-Objekt (OBJ_EDIT)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | <a href="#">ENUM_ALIGN_MODE</a>                            |
| OBJPROP_FONTSIZE   | Schriftgröße                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | int                                                        |

|                    |                                                                                                                                                                                                                                                                      |                                                                                                                                        |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------|
| OBJPROP_RAY_LEFT   | Strahl setzt sich nach links fort                                                                                                                                                                                                                                    | bool                                                                                                                                   |
| OBJPROP_RAY_RIGHT  | Strahl setzt sich nach rechts fort                                                                                                                                                                                                                                   | bool                                                                                                                                   |
| OBJPROP_RAY        | Eine vertikale Linie erstreckt sich auf alle der Chart-Fenster                                                                                                                                                                                                       | bool                                                                                                                                   |
| OBJPROP_ELLIPSE    | Anzeigen die ganzen Ellipse für das Objekt "Fibonacci Arcs" ( <a href="#">OBJ_FIBOARC</a> )                                                                                                                                                                          | bool                                                                                                                                   |
| OBJPROP_ARROWCODE  | Weichenkode für Objekt "Weiche"                                                                                                                                                                                                                                      | char                                                                                                                                   |
| OBJPROP_TIMEFRAMES | Sichtbarkeit des Objekts auf Timeframes                                                                                                                                                                                                                              | Flaggenvorrat <a href="#">flags</a>                                                                                                    |
| OBJPROP_ANCHOR     | Stellung des Bezugspunktes des graphischen Objekts                                                                                                                                                                                                                   | <a href="#">ENUM_ARROW_ANCHOR</a> (für OBJ_ARROW),<br><a href="#">ENUM_ANCHOR_POINT</a> (für OBJ_LABEL, OBJ_BITMAP_LABEL und OBJ_TEXT) |
| OBJPROP_XDISTANCE  | Distanz in Pixel X-Achse vom Bezugswinkel (см. <a href="#">примечание</a> )                                                                                                                                                                                          | int                                                                                                                                    |
| OBJPROP_YDISTANCE  | Distanz in Pixel Y-Achse vom Bezugswinkel (см. <a href="#">примечание</a> )                                                                                                                                                                                          | int                                                                                                                                    |
| OBJPROP_DIRECTION  | Trend des Gann Objekts                                                                                                                                                                                                                                               | <a href="#">ENUM_GANN_DIRECTION</a>                                                                                                    |
| OBJPROP_DEGREE     | Level von Elliott Wave Markierung                                                                                                                                                                                                                                    | <a href="#">ENUM_ELLIOT_WAVE_DEGREE</a>                                                                                                |
| OBJPROP_DRAWLINES  | Liniendarstellung für Elliott Wave Markierung                                                                                                                                                                                                                        | bool                                                                                                                                   |
| OBJPROP_STATE      | Schaltflächenstellung (Gedrückt/nicht gedrückt)                                                                                                                                                                                                                      | bool                                                                                                                                   |
| OBJPROP_CHART_ID   | ID von Objekt "Chart" ( <a href="#">OBJ_CHART</a> ). Es erlaubt mit den Eigenschaften dieses Objektes als einer regelmäßigen Diagram mit den Funktionen der Sektion <a href="#">Operationen mit Charts</a> arbeiten, aber es gibt einige <a href="#">Ausnahmen</a> . | long r/o                                                                                                                               |
| OBJPROP_XSIZE      | Die Breite des Objekts entlang der X-Achse in Pixeln. Wird für                                                                                                                                                                                                       | int                                                                                                                                    |

|                     |                                                                                                                                                                                                                                                                          |                                  |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------|
|                     | OBJ_LABEL (read only),<br>OBJ_BUTTON, OBJ_CHART,<br>OBJ_BITMAP,<br>OBJ_BITMAP_LABEL,<br>OBJ_EDIT,<br>OBJ_RECTANGLE_LABEL<br>angegeben.                                                                                                                                   |                                  |
| OBJPROP_YSIZE       | Die Höhe des Objekts entlang der Y-Achse in Pixeln. Wird für OBJ_LABEL (read only), OBJ_BUTTON, OBJ_CHART, OBJ_BITMAP, OBJ_BITMAP_LABEL, OBJ_EDIT, OBJ_RECTANGLE_LABEL angegeben.                                                                                        | int                              |
| OBJPROP_XOFFSET     | Die X Koordinate von der linken oberen Ecke des <u>rechteckigen sichtbaren Bereich</u> in der grafischen Objekte "Bitmap Label" und "Bitmap" (OBJ_BITMAP_LABEL und OBJ_BITMAP). Der Wert wird in Pixel relativ zur oberen linken Ecke des ursprünglichen Bildes gesetzt. | int                              |
| OBJPROP_YOFFSET     | Die Y Koordinate von der linken oberen Ecke des <u>rechteckigen sichtbaren Bereich</u> in der grafischen Objekte "Bitmap Label" und "Bitmap" (OBJ_BITMAP_LABEL und OBJ_BITMAP). Der Wert wird in Pixel relativ zur oberen linken Ecke des ursprünglichen Bildes gesetzt. | int                              |
| OBJPROP_PERIOD      | Periode für Objekt "Chart"                                                                                                                                                                                                                                               | <a href="#">ENUM_TIMEFRAMES</a>  |
| OBJPROP_DATE_SCALE  | Merkmal der Darstellung von Preisskala für Objekt "Chart"                                                                                                                                                                                                                | bool                             |
| OBJPROP_PRICE_SCALE | Merkmal der Darstellung von Preisskala für Objekt "Chart"                                                                                                                                                                                                                | bool                             |
| OBJPROP_CHART_SCALE | Masstab für Objekt "Chart"                                                                                                                                                                                                                                               | int Größe im Bereich 0-5         |
| OBJPROP_BGCOLOR     | Hintergrundfarbe für OBJ_EDIT, OBJ_BUTTON, OBJ_RECTANGLE_LABEL                                                                                                                                                                                                           | color                            |
| OBJPROP_CORNER      | Winkel des Charts für Snap                                                                                                                                                                                                                                               | <a href="#">ENUM_BASE_CORNER</a> |

|                      |                                                         |                                  |
|----------------------|---------------------------------------------------------|----------------------------------|
|                      | des Graphischen Objektes                                |                                  |
| OBJPROP_BORDER_TYPE  | Frame-Typ für Object "Rechteckige Marke"                | <a href="#">ENUM_BORDER_TYPE</a> |
| OBJPROP_BORDER_COLOR | Die Rahmenfarbe für die Objekte OBJ_EDIT und OBJ_BUTTON | color                            |

Wenn Sie benutzen [Operationen mit Charts](#) für Objekt "Chart" ([OBJ\\_CHART](#)), hat es die folgenden Einschränkungen:

- Es kann nicht durch [ChartClose\(\)](#) geschlossen werden;
- Symbol/Periode kann nicht durch [ChartSetSymbolPeriod\(\)](#) geändert werden;
- Die folgenden Eigenschaften funktionieren nicht: CHART\_SCALE, CHART\_BRING\_TO\_TOP, CHART\_SHOW\_DATE\_SCALE und CHART\_SHOW\_PRICE\_SCALE ([ENUM\\_CHART\\_PROPERTY\\_INTEGER](#)).

Für Objekte [OBJ\\_BITMAP\\_LABEL](#) und [OBJ\\_BITMAP](#), kann ein besonderer Modus der Bilddarstellung festgelegt werden. In diesem Modus nur ein Teil des Bildes, bei dem eine rechteckige sichtbaren Bereich angewendet wird, während der Rest des Bildes wird unsichtbar. Die Größe dieses Bereiches sollten mit den Eigenschaften OBJPROP\_XSIZE und OBJPROP\_YSIZE gestellt werden. Der sichtbare Bereich kann nur innerhalb des ursprünglichen Bildes mit den Eigenschaften OBJPROP\_XOFFSET und OBJPROP\_YOFFSET "bewegt" werden.

Для объектов с фиксированными размерами: [OBJ\\_BUTTON](#), [OBJ\\_RECTANGLE\\_LABEL](#), [OBJ\\_EDIT](#) и [OBJ\\_CHART](#) свойства OBJPROP\_XDISTANCE и OBJPROP\_YDISTANCE задают положение левой верхней точки объекта относительно угла графика (OBJPROP\_CORNER), от которого будут отсчитываться координаты X и Y в пикселях.

für die Funktionen [ObjectSetDouble\(\)](#) und [ObjectGetDouble\(\)](#)

#### ENUM\_OBJECT\_PROPERTY\_DOUBLE

| Identifikator      | Beschreibung                                                                                                  | Typ der Eigenschaft                         |
|--------------------|---------------------------------------------------------------------------------------------------------------|---------------------------------------------|
| OBJPROP_PRICE      | Koordinate des Preises                                                                                        | double Modifikator=Nummer des Bezugspunktes |
| OBJPROP_LEVELVALUE | Levelwert                                                                                                     | double Modifikator=Levelnummer              |
| OBJPROP_SCALE      | Massstab (Eigenschaft der Objekte des Objektes "Fibonacci Arcs")                                              | double                                      |
| OBJPROP_ANGLE      | Winkel. Für die aus einem Programm erstellten Objekte, für die kein Winkel angegeben ist, ist der Wert gleich | double                                      |

|                   |                                             |        |
|-------------------|---------------------------------------------|--------|
|                   | <a href="#">EMPTY_VALUE</a>                 |        |
| OBJPROP_DEVIATION | Abweichung für Kanal der Standardabweichung | double |

für die Funktionen [ObjectSetString\(\)](#) und [ObjectGetString\(\)](#)

#### ENUM\_OBJECT\_PROPERTY\_STRING

| Identifikator      | Beschreibung                                                                                                                                                                                             | Typ der Eigenschaft                               |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------|
| OBJPROP_NAME       | Objektname                                                                                                                                                                                               | string                                            |
| OBJPROP_TEXT       | Objektbeschreibung (Text, der im Objekt gibt)                                                                                                                                                            | string                                            |
| OBJPROP_TOOLTIP    | Der Text eines Tooltip. Wenn die Eigenschaft nicht angegeben ist, wird der Tooltip automatisch vom Terminal generiert. Ein Tooltip kann durch Setzen des "\n" (Zeilenvorschub) Wertes deaktiviert werden | string                                            |
| OBJPROP_LEVELTEXT  | Levelbeschreibung                                                                                                                                                                                        | string<br>Modifikator=Levelnummer                 |
| OBJPROP_FONT       | Schriftart                                                                                                                                                                                               | string                                            |
| OBJPROP_BITMAPFILE | Name der BMP-Datei für Objekt "Graphische Marke". Sehen Sie auch <a href="#">Ressourcen</a>                                                                                                              | string Modifikator: 0-Stellung ON, 1-Stellung OFF |
| OBJPROP_SYMBOL     | Symbol für Objekt "Chart"                                                                                                                                                                                | string                                            |

Für OBJ\_RECTANGLE\_LABEL ("rechteckige Marke") kann Mann eine von drei Arten, die auf Werte im ENUM\_BORDER\_TYPE entsprechen, angeben.

#### ENUM\_BORDER\_TYPE

| Identifikator | Beschreibung |
|---------------|--------------|
| BORDER_FLAT   | Flache Form  |
| BORDER_RAISED | Konvexe Form |
| BORDER_SUNKEN | Konkave Form |

Für das Objekt OBJ\_EDIT ("Eingabefeld") und die Funktion [ChartScreenShot\(\)](#) können Sie den Typ der horizontalen Ausrichtung mit der Enumerationswerte ENUM\_ALIGN\_MODE angeben.

## ENUM\_ALIGN\_MODE

| Identifizier | Beschreibung                               |
|--------------|--------------------------------------------|
| ALIGN_LEFT   | Linksbündige Ausrichtung                   |
| ALIGN_CENTER | Zentrierte Ausrichtung (für "Eingabefeld") |
| ALIGN_RIGHT  | Rechtsbündige Ausrichtung                  |

## Beispiel:

```
#define UP          "\x0431"

//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//---
    string label_name="my_OBJ_LABEL_object";
    if(ObjectFind(0,label_name)<0)
    {
        Print("Object",label_name,"not found. Error code = ",GetLastError());
        //--- erzeugen wir das Objekt Label
        ObjectCreate(0,label_name,OBJ_LABEL,0,0,0);
        //--- stellen wir die Koordinate X ein
        ObjectSetInteger(0,label_name,OBJPROP_XDISTANCE,200);
        //--- stellen wir die Koordinate Y ein
        ObjectSetInteger(0,label_name,OBJPROP_YDISTANCE,300);
        //--- Geben wir Textfarbe vor
        ObjectSetInteger(0,label_name,OBJPROP_COLOR,clrWhite);
        //--- stellen wir text für Objekt Label ein
        ObjectSetString(0,label_name,OBJPROP_TEXT,UP);
        //--- stellen wir Schrift der Unterschrift ein
        ObjectSetString(0,label_name,OBJPROP_FONT,"Wingdings");
        //--- Stellen wir Schriftgroesse ein
        ObjectSetInteger(0,label_name,OBJPROP_FONTSIZE,10);
        //--- drehen wir um 45 Grad mit dem Uhrzeigerlauf herum
        ObjectSetDouble(0,label_name,OBJPROP_ANGLE,-45);
        //--- verbieten wir Hervorhebung mit der Maus
        ObjectSetInteger(0,label_name,OBJPROP_SELECTABLE,false);
        //--- zeichnen wir auf dem Chart
        ChartRedraw(0);
    }
};
```

## Methode der Objektbindung

Graphische Objekte Text und Label (OBJ\_TEXT, OBJ\_BITMAP\_LABEL und OBJ\_LABEL) können einen der 9 verschiedenen Wege der Koordinatenverbindung haben. Die notwendige Variante kann durch die Funktion `ObjectSetInteger(handle_Chart, Name_Objekt, OBJPROP_ANCHOR, Verbindungs_Weg)` angegeben werden, wo Verbindungs\_Weg - einer der Werte von Aufzählung ENUM\_ANCHOR\_POINT ist.

### ENUM\_ANCHOR\_POINT

| Identifikator      | Beschreibung                         |
|--------------------|--------------------------------------|
| ANCHOR_LEFT_UPPER  | Binde im oberen linken Winkel        |
| ANCHOR_LEFT        | Bindepunkt links zentriert           |
| ANCHOR_LEFT_LOWER  | Bindepunkt im unteren linken Winkel  |
| ANCHOR_LOWER       | Bindepunkt unten zentriert           |
| ANCHOR_RIGHT_LOWER | Bindepunkt im unteren rechten Winkel |
| ANCHOR_RIGHT       | Bindepunkt rechts zentriert          |
| ANCHOR_RIGHT_UPPER | Bindepunkt rechts zentriert          |
| ANCHOR_UPPER       | Bindepunkt oben zentriert            |
| ANCHOR_CENTER      | Bindepunkt im Objektzentrum          |

Объекты `OBJ_BUTTON`, `OBJ_RECTANGLE_LABEL`, `OBJ_EDIT` и `OBJ_CHART` имеют фиксированную точку привязки в левом верхнем углу (ANCHOR\_LEFT\_UPPER).

### Beispiel:

```
string text_name="my_OBJ_TEXT_object";
if(ObjectFind(0,text_name)<0)
{
    Print("Object",text_name,"not found. Error code = ",GetLastError());
    //--- wir bekommen maximalen Preis des Charts
    double chart_max_price=ChartGetDouble(0,CHART_PRICE_MAX,0);
    //--- erzeugen wir das Objekt Label
    ObjectCreate(0,text_name,OBJ_TEXT,0,TimeCurrent(),chart_max_price);
    //--- stellen wir Textfarbe fest
    ObjectSetInteger(0,text_name,OBJPROP_COLOR,clrWhite);
    //--- stellen wir Hintergrundfarbe fest
    ObjectSetInteger(0,text_name,OBJPROP_BGCOLOR,clrGreen);
    //--- stellen wir Text für Objekt Label fest
    ObjectSetString(0,text_name,OBJPROP_TEXT,TimeToString(TimeCurrent()));
    //--- stellen wir Schriftgroesse der überschrift fest
    ObjectSetString(0,text_name,OBJPROP_FONT,"Trebuchet MS");
    //--- stellen wir Schriftgroesse fest
    ObjectSetInteger(0,text_name,OBJPROP_FONTSIZE,10);
    //--- stellen wir snap zum oberen rechten Winkel fest
    ObjectSetInteger(0,text_name,OBJPROP_ANCHOR,ANCHOR_RIGHT_UPPER);
}
```



```

//--- schwenken wir um 90 Grad gegen dem Uhrzeigerlauf
ObjectSetDouble(0,text_name,OBJPROP_ANGLE,90);
//--- verbieten wir Objekthervorheben mit der Maus
ObjectSetInteger(0,text_name,OBJPROP_SELECTABLE,false);
//--- zeichnen wir auf dem Chart
ChartRedraw(0);
; }

```

Graphische Objekte Arrow (OBJ\_ARROW) haben nur 2 Snapverfahren ihrer Koordinaten. Identifikatoren werden in ENUM\_ARROW\_ANCHOR aufgezählt.

#### ENUM\_ARROW\_ANCHOR

| Identifikator | Beschreibung                              |
|---------------|-------------------------------------------|
| ANCHOR_TOP    | Bindepunkt für Weiche befindet sich oben  |
| ANCHOR_BOTTOM | Bindepunkt für Weiche befindet sich unten |

#### Beispiel:

```

void OnStart()
{
//--- Dienstfelder
double Ups[],Downs[];
datetime Time[];
//--- stellen wir für Felder Timeserienmerkmal fest
ArraySetAsSeries(Ups,true);
ArraySetAsSeries(Downs,true);
ArraySetAsSeries(Time,true);
//--- erzeugen wir handle des Indikators Fractals
int FractalsHandle=iFractals(NULL,0);
Print("FractalsHandle =",FractalsHandle);
//--- stellen wir Fehlerkode auf Null
ResetLastError();
//--- versuchen wir, Indikatorwert zu kopieren
int copied=CopyBuffer(FractalsHandle,0,0,1000,Ups);
if(copied<=0)
{
Print("Obere Fractals Kopieren Fehler. Error = ",GetLastError());
return;
}

ResetLastError();
//--- versuchen wir Indikatorwerte zu kopieren
copied=CopyBuffer(FractalsHandle,1,0,1000,Downs);
if(copied<=0)
{
Print("Untere Fractals Kopieren Fehler. Error = ",GetLastError());
return;
}
; }

```

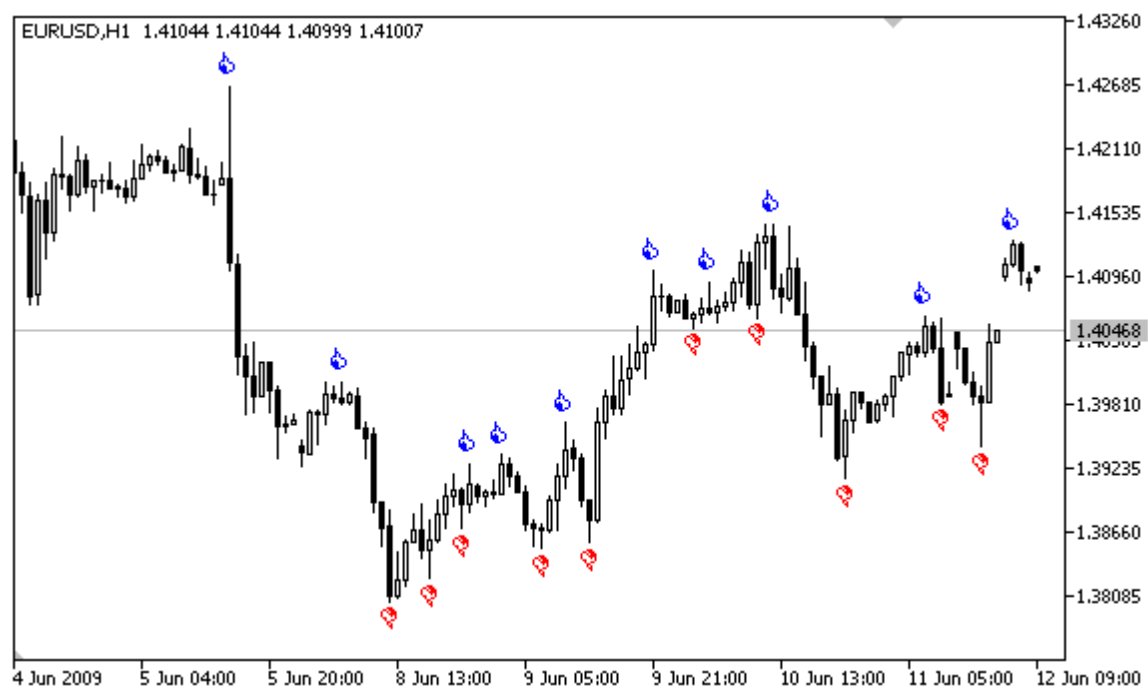
```

ResetLastError();
//--- kopieren wir Zeitreihe mit der Eröffnungszeit der letzten 1000 Bars
copied=CopyTime(NULL,0,0,1000,Time);
if(copied<=0)
{
    Print("Nicht gelungen, Eröffnungszeit der letzten 1000 Bars zu kopieren");
    return;
};

int upcounter=0,downcounter=0; // zählen wir die Weichen
bool created; // bekommen wir das Ergebnis des Versuches, ein Objekt zu erzeugen
for(int i=2;i<copied;i++) // sehen wir die Werte des Indikators iFractals durch
{
    if(Ups[i]!=EMPTY_VALUE) // oberer Fraktal gefunden
    {
        if(upcounter<10) // erzeugen nicht mehr als 10 Objekte "oben"
        {
            //--- versuchen wir das Objekt "oben" zu erzeugen
            created=ObjectCreate(0,string(Time[i]),OBJ_ARROW_THUMB_UP,0,Time[i],Ups[i]);
            if(created) // wenn erzeugt - machen wir tuning dafür
            {
                //--- Bindepunkt ist unten, um nicht um Bar nicht zu bedecken
                ObjectSetInteger(0,string(Time[i]),OBJPROP_ANCHOR,ANCHOR_BOTTOM);
                //--- letzter Strich - zeichnen
                ObjectSetInteger(0,string(Time[i]),OBJPROP_COLOR,clrBlue);
                upcounter++;
            };
        };
    };
    if(Downs[i]!=EMPTY_VALUE) // neuer Fraktal gefunden
    {
        if(downcounter<10) // erzeugen wir nicht mehr als 10 Objekte "unten"
        {
            //--- versuchen wir das Objekt "unten" zu erzeugen
            created=ObjectCreate(0,string(Time[i]),OBJ_ARROW_THUMB_DOWN,0,Time[i],Downs[i]);
            if(created) // wenn erzeugt - machen wir tuning dafür
            {
                //--- Bindepunkt ist oben, um die Bar nicht zu bedecken
                ObjectSetInteger(0,string(Time[i]),OBJPROP_ANCHOR,ANCHOR_TOP);
                //--- letzter Strich - zeichnen
                ObjectSetInteger(0,string(Time[i]),OBJPROP_COLOR,clrRed);
                downcounter++;
            };
        };
    };
};
};
};

```

Nach der Durchführung von Script wird der Chart etwa so aussehen, wie auf diesem Bild.



## Chartwinkel, zu dem Objekt gebunden ist

Es gibt eine Reihe von graphischen Objekten, für die der Chartwinkel festgelegt werden kann, dem gegenüber Koordinaten in Pixel angegeben werden. Das sind folgende Objekttypen (in Klammern werden Identifikatoren des Objekttyps angegeben):

- Label (OBJ\_LABEL);
- Button (OBJ\_BUTTON);
- Chart (OBJ\_CHART);
- Bitmap Label (OBJ\_BITMAP\_LABEL);
  - Rectangle Label (OBJ\_RECTANGLE\_LABEL);
- Edit (OBJ\_EDIT).

Um den Chartwinkel anzugeben, von dem Koordinaten X und Y in Pixel abgezählt werden, muss man die Funktion ObjectSetInteger(chartID, name, OBJPROP\_CORNER, chart\_corner) verwenden, wo:

- chartID - Chartidentifikator;
- name - Name des graphischen Objekts;
- OBJPROP\_CORNER - Identifikator der Eigenschaft für Festlegung des Snapwinkels;
- chart\_corner - der notwendiger Chartwinkel kann einen der Werte von Aufzählung ENUM\_BASE\_CORNER annehmen.

### ENUM\_BASE\_CORNER

| Identifikator      | Beschreibung                                                     |
|--------------------|------------------------------------------------------------------|
| CORNER_LEFT_UPPER  | Mittelpunkt der Koordinaten im oberen linken Winkel des Charts   |
| CORNER_LEFT_LOWER  | Mittelpunkt der Koordinaten im unteren linken Winkel des Charts  |
| CORNER_RIGHT_LOWER | Mittelpunkt der Koordinaten im unteren rechten Winkel des Charts |
| CORNER_RIGHT_UPPER | Mittelpunkt der Koordinaten im oberen rechten Winkel des Charts  |

### Beispiel:

```
void CreateLabel(long   chart_id,
                string  name,
                int     chart_corner,
                string  text_label,
                int     x_ord,
                int     y_ord)
{
    //---
    ObjectCreate(chart_id,name,OBJ_LABEL,0,0,0);
    ResetLastError();
    if(!ObjectSetInteger(chart_id,name,OBJPROP_CORNER,chart_corner))
        Print("Snapwinkel für Objekt einstellen Fehler ",
```

```

        name, " Fehler ", GetLastError());
    ObjectSetInteger(chart_id,name,OBJPROP_XDISTANCE,x_ord);
    ObjectSetInteger(chart_id,name,OBJPROP_YDISTANCE,y_ord);
    ObjectSetString(chart_id,name,OBJPROP_TEXT,text_label);
; }
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//---
    int height=ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0);
    int width=ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0);
    string arrows[4]={"LEFT_UPPER","RIGHT_UPPER","RIGHT_LOWER","LEFT_LOWER"};
    CreateLabel(0,arrows[0],CORNER_LEFT_UPPER,"0",50,50);
    CreateLabel(0,arrows[1],CORNER_RIGHT_UPPER,"1",50,50);
    CreateLabel(0,arrows[2],CORNER_RIGHT_LOWER,"2",50,50);
    CreateLabel(0,arrows[3],CORNER_LEFT_LOWER,"3",50,50);
; }

```

## Objektsichtbarkeit

Kombination von Flaggen der Objektsichtbarkeit destimmt Timeframes des Charts, auf denen das Objekt dargestellt wird. Für Einstellung/Erhaltung des Wertes der Eigenschaft OBJPROP\_TIMEFRAMES können die Funktionen [ObjectSetInteger\(\)](#)/[ObjectGetInteger\(\)](#) verwendet werden.

| Konstante      | Wert       | Beschreibung                                |
|----------------|------------|---------------------------------------------|
| OBJ_PERIOD_M1  | 0x00000001 | Objekt wird auf 1-Minute Chart gezeichnet   |
| OBJ_PERIOD_M2  | 0x00000002 | Objekt wird auf 2-Minuten Chart gezeichnet  |
| OBJ_PERIOD_M3  | 0x00000004 | Objekt wird auf 3-Minuten Chart gezeichnet  |
| OBJ_PERIOD_M4  | 0x00000008 | Objekt wird auf 4-Minuten Chart gezeichnet  |
| OBJ_PERIOD_M5  | 0x00000010 | Objekt wird auf 5-Minuten Chart gezeichnet  |
| OBJ_PERIOD_M6  | 0x00000020 | Objekt wird auf 6-Minuten Chart gezeichnet  |
| OBJ_PERIOD_M10 | 0x00000040 | Objekt wird auf 10-Minuten Chart gezeichnet |
| OBJ_PERIOD_M12 | 0x00000080 | Objekt wird auf 12-Minuten Chart gezeichnet |
| OBJ_PERIOD_M15 | 0x00000100 | Objekt wird auf 15-Minuten Chart gezeichnet |
| OBJ_PERIOD_M20 | 0x00000200 | Objekt wird auf 20-Minuten Chart gezeichnet |
| OBJ_PERIOD_M30 | 0x00000400 | Objekt wird auf 30-Minuten Chart gezeichnet |
| OBJ_PERIOD_H1  | 0x00000800 | Objekt wird auf 1-Stunde Chart gezeichnet   |
| OBJ_PERIOD_H2  | 0x00001000 | Objekt wird auf 2-Stunden Chart gezeichnet  |
| OBJ_PERIOD_H3  | 0x00002000 | Objekt wird auf 3-Stunden Chart gezeichnet  |
| OBJ_PERIOD_H4  | 0x00004000 | Objekt wird auf 4-Stunden Chart gezeichnet  |
| OBJ_PERIOD_H6  | 0x00008000 | Objekt wird auf 6-Stunden Chart gezeichnet  |
| OBJ_PERIOD_H8  | 0x00010000 | Objekt wird auf 8-Stunden Chart gezeichnet  |

|                 |            |                                             |
|-----------------|------------|---------------------------------------------|
| OBJ_PERIOD_H12  | 0x00020000 | Objekt wird auf 12-Stunden Chart gezeichnet |
| OBJ_PERIOD_D1   | 0x00040000 | Objekt wird auf Tageschart gezeichnet       |
| OBJ_PERIOD_W1   | 0x00080000 | Objekt wird auf Wochenchart gezeichnet      |
| OBJ_PERIOD_MN1  | 0x00100000 | Objekt wird auf Monatschart gezeichnet      |
| OBJ_ALL_PERIODS | 0x001fffff | Objekt wird auf allen Timeframes gezeichnet |

Flaggen der Sichtbarkeit können durch das Symbol "|" kombiniert werden, zB Flaggenkombination OBJ\_PERIOD\_M10|OBJ\_PERIOD\_H4 bedeutet, dass Objekt auf dem 10-Minuten und 4-Minuten timeframes sichtbar sein wird.

#### Beispiel:

```
void OnStart()
{
    //---
    string highlevel="PreviousDayHigh";
    string lowlevel="PreviousDayLow";
    double prevHigh;           // High des Vortages
    double prevLow;            // Low des Vortages
    double highs[], lows[];    // Felder für Erhaltung von High und Low

    //--- stellen wir den Wert des letzten Fehlers zurück
    ResetLastError();
    //--- wir bekommen 2 letzte Werte High auf Tagst timeframe
    int highsgot=CopyHigh(Symbol(), PERIOD_D1, 0, 2, highs);
    if(highsgot>0) // wenn Kopieren erfolgreich ist
    {
        Print("Preise High für letzte 2 Tage erfolgreich erhalten");
        prevHigh=highs[0]; // High des Vortages
        Print("prevHigh =", prevHigh);
        if(ObjectFind(0, highlevel)<0) // Objekt mit dem Namen highlevel nicht gefunden
        {
            ObjectCreate(0, highlevel, OBJ_HLINE, 0, 0, 0); // schaffen wir das Objekt Hotline
        }
        //--- stellen wir Preislevel für die Linie highlevel fest
        ObjectSetDouble(0, highlevel, OBJPROP_PRICE, 0, prevHigh);
        //--- Stellen wir Sichtbarkeit nur für PERIOD_M10 und PERIOD_H4
        ObjectSetInteger(0, highlevel, OBJPROP_TIMEFRAMES, OBJ_PERIOD_M10|OBJ_PERIOD_H4);
    }
    else
    {
        Print("Erhalten von Preisen High für letzte 2 Tage Fehler, Error = ", GetLastError());
    }
}
```

```

//---stellen wir den wert des letzten Fehlers zurück
ResetLastError();
//--- bekommen wir 2 letzte Werte auf Tagstimeframe Low
int lowsgot=CopyLow(Symbol(),PERIOD_D1,0,2, lows);
if(lowsgot>0) // wenn Kopierenoperation erfolgreich ist
    Print("Preise Low für letzte 2 Tage erfolgreich erhalten");
prevLow=lows[0]; // Low des Vortages
Print("prevLow =",prevLow);
if(ObjectFind(0,lowlevel)<0) // Objekt mit dem Namen lowlevel nicht gefunden
{
    ObjectCreate(0,lowlevel,OBJ_HLINE,0,0,0); // erzeugen wir das Objekt "Hotline
; }
//--- stellen wir Preislevel für Linie lowlevel fest
ObjectSetDouble(0,lowlevel,OBJPROP_PRICE,0,prevLow);
//--- stellen wir Sichtbarkeit nur für PERIOD_M10 und PERIOD_H4 fest
ObjectSetInteger(0,lowlevel,OBJPROP_TIMEFRAMES,OBJ_PERIOD_M10|OBJ_PERIOD_H4);
; }
else Print("Erhalten von Preisen für die letzten zwei Tage Fehler, Error = ",GetLas

ChartRedraw(0); // zeichnen wir Chart zwangsläufig neu
; }

```

Sehen Sie auch

[PeriodSeconds](#), [Period](#), [Chartperioden](#), [Datum und Zeit](#)



## Levels von Elliott Wellen

Elliott Wellen werden durch zwei graphische Objekte der Typen OBJ\_ELLIOTWAVE5 und OBJ\_ELLIOTWAVE3 dargestellt. Für Festlegung der Wellengroesse (Verfahren der Wellenmarkierung) wird die Eigenschaft OBJPROP\_DEGREE verwendet, der einen der Aufzählungswerte ENUM\_ELLIOT\_WAVE\_DEGREE festgestellt werden kann.

### ENUM\_ELLIOT\_WAVE\_DEGREE

| Konstante                | Beschreibung                        |
|--------------------------|-------------------------------------|
| ELLIOTT_GRAND_SUPERCYCLE | Hauptsuperzyklus (Grand Supercycle) |
| ELLIOTT_SUPERCYCLE       | Superzyklus (Supercycle)            |
| ELLIOTT_CYCLE            | Zyklus (Cycle)                      |
| ELLIOTT_PRIMARY          | primaeres Zyklus (Primary)          |
| ELLIOTT_INTERMEDIATE     | Mittelstelle (Intermediate)         |
| ELLIOTT_MINOR            | Sekundaeres Zyklus (Minor)          |
| ELLIOTT_MINUTE           | Minute (Minute)                     |
| ELLIOTT_MINUETTE         | Sekunde (Minuette)                  |
| ELLIOTT_SUBMINUETTE      | Subsekunde (Subminuette)            |

### Beispiel:

```
for(int i=0;i<ObjectsTotal(0);i++)
{
    string currobj=ObjectName(0,i);
    if((ObjectGetInteger(0,currobj,OBJPROP_TYPE)==OBJ_ELLIOTWAVE3) ||
        ((ObjectGetInteger(0,currobj,OBJPROP_TYPE)==OBJ_ELLIOTWAVE5)))
    {
        //--- stellen wir Einteilungsstufe in INTERMEDIATE
        ObjectSetInteger(0,currobj,OBJPROP_DEGREE,ELLIOTT_INTERMEDIATE);
        //--- Zeigen von Linien unter Wellenbögen schalten
        ObjectSetInteger(0,currobj,OBJPROP_DRAWLINES,true);
        //--- stellen wir Farbe der Linien ein
        ObjectSetInteger(0,currobj,OBJPROP_COLOR,clrBlue);
        //--- stellen wir Dicke der Linien ein
        ObjectSetInteger(0,currobj,OBJPROP_WIDTH,5);
        //--- stellen wir Beschreibung fest
        ObjectSetString(0,currobj,OBJPROP_TEXT,"test script");
    }
};
```

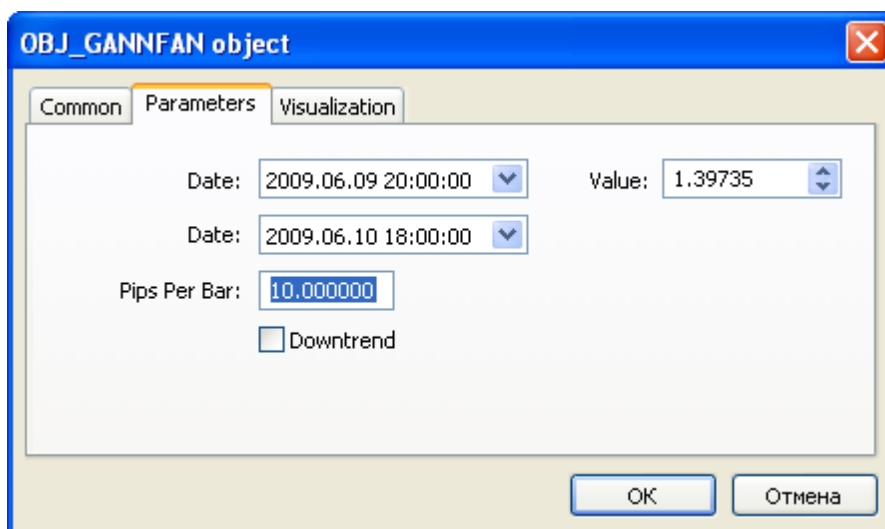
## Gann Objekte

für Objekte Gannfan (OBJ\_GANNFAN) und Ganngrid (OBJ\_GANNGRID) kann einer der zwei Werte der Aufzählung ENUM\_GANN\_DIRECTION angegeben werden, der Trendrichtung feststellt.

| Konstante       | Beschreibung                          |
|-----------------|---------------------------------------|
| GANN_UP_TREND   | Linie entspricht dem steigenden Trend |
| GANN_DOWN_TREND | Linie entspricht dem fallenden Trend  |

für Einstellung des Maßstabes der Grundlinie 1x1 wird die Funktion [ObjectSetDouble](#)(chart\_handle, gann\_object\_name, OBJPROP\_SCALE, scale) verwendet, wo:

- chart\_handle - Chartfenster, in dem sich Objekt befindet;
- gann\_object\_name - Objektname;
- OBJPROP\_SCALE - Identifikator der Eigenschaft "Scale";
- scale - der notwendige Maßstab in Einheiten Pips/Bar.



Beispiel der Erzeugung von Gannfan:

```
void OnStart()
{
//---
string my_gann="OBJ_GANNFAN object";
if(ObjectFind(0,my_gann)<0) // Objekt nicht gefunden
{
//--- melden wir Misserfolg an
Print("Object ",my_gann," not found. Error code = ",GetLastError());
//--- wir erhalten maximalen Preis des Charts
double chart_max_price=ChartGetDouble(0,CHART_PRICE_MAX,0);
//--- wir erhalten minimalen Preis des Charts
double chart_min_price=ChartGetDouble(0,CHART_PRICE_MIN,0);
//--- wieviel Bars auf dem Chart gezeigt?
int bars_on_chart=ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- erzeugen wir Feld, wo wir die Eröffnungszeit jeder Bar schreiben
```

```

datetime Time[];
//--- organisieren wir Zugang zum Feld wie in Timeserie
ArraySetAsSeries(Time,true);
//--- Jetzt kopieren wir darin Daten für die auf dem Chart sichtbaren Bars
int times=CopyTime(NULL,0,0,bars_on_chart,Time);
if(times<=0)
{
    Print("Kopieren des Feldes mit Eröffnungszeit fehler!");
    return;
};
//--- Vorbereitungen werden abgeschlossen

//--- Index der Zentralbar auf dem Chart
int center_bar=bars_on_chart/2;
//--- Chartäquator - zwischen Maximum und Minimum
double mean=(chart_max_price+chart_min_price)/2.0;
//--- stellen wir Koordinaten des ersten Snappunktes ins Zentrum fest
ObjectCreate(0,my_gann,OBJ_GANNFAN,0,Time[center_bar],mean,
    //--- der zweite Snappunkt ist rechts
    Time[center_bar/2],(mean+chart_min_price)/2.0);
Print("Time[center_bar]/="+string(Time[center_bar])+" Time[center_bar/2]="+string(Time[center_bar/2]));
//Print("Time[center_bar]/="+Time[center_bar]+" Time[center_bar/2]="+Time[center_bar/2]);
//--- stellen wir Massstab in Einheiten Pips/Bar
ObjectSetDouble(0,my_gann,OBJPROP_SCALE,10);
//--- stellen wir Liniendicke fest
ObjectSetInteger(0,my_gann,OBJPROP_DIRECTION,GANN_UP_TREND);
//--- stellen wir Liniendicke fest
ObjectSetInteger(0,my_gann,OBJPROP_WIDTH,1);
//--- Legen wir Linienstil
ObjectSetInteger(0,my_gann,OBJPROP_STYLE,STYLE_DASHDOT);
//--- und Linienfarbe fest
ObjectSetInteger(0,my_gann,OBJPROP_COLOR,clrYellowGreen);
//--- erlauben wir dem Benutzer, Objekt hervorzuheben
ObjectSetInteger(0,my_gann,OBJPROP_SELECTABLE,true);
//--- herben wir es selbst hervor
ObjectSetInteger(0,my_gann,OBJPROP_SELECTED,true);
//--- zeichnen wir auf dem Chart
ChartRedraw(0);
};
};

```

## Web-Farben

für Typ [color](#) werden folgende Preiskonstanten bestimmt:

|                  |                      |                    |                     |                  |                 |                   |                   |
|------------------|----------------------|--------------------|---------------------|------------------|-----------------|-------------------|-------------------|
|                  |                      |                    |                     |                  |                 |                   |                   |
|                  |                      |                    |                     |                  |                 |                   |                   |
|                  |                      |                    |                     |                  |                 | clrDarkTurquoise  |                   |
| clrLightSeaGreen |                      |                    |                     |                  |                 |                   |                   |
| clrGoldenrod     | clrMediumSpringGreen | clrLawnGreen       |                     |                  |                 |                   |                   |
|                  | clrOrange            | clrGold            | clrYellow           | clrChartreuse    | clrLime         | clrSpringGreen    | clrAqua           |
| clrDeepSkyBlue   |                      | clrMagenta         |                     |                  |                 |                   |                   |
|                  |                      | clrMediumTurquoise |                     | clrTurquoise     |                 |                   | clrDarkKhaki      |
|                  |                      | clrGreenYellow     | clrMediumAquamarine |                  |                 |                   | clrOrchid         |
| clrMediumPurple  |                      |                    |                     | clrDarkGray      | clrSandyBrown   |                   | clrTan            |
|                  | clrBurlyWood         | clrHotPink         |                     | clrViolet        |                 | clrSkyBlue        | clrLightSalmon    |
| clrPlum          | clrKhaki             | clrLightGreen      | clrAquamarine       | clrSilver        | clrLightSkyBlue | clrLightSteelBlue | clrLightBlue      |
| clrPaleGreen     | clrThistle           | clrPowderBlue      | clrPaleGoldenrod    | clrPaleTurquoise | clrLightGray    | clrWheat          | clrNavajoWhite    |
| clrMoccasin      | clrLightPink         | clrGainsboro       | clrPeachPuff        | clrPink          | clrBisque       | clrLightGoldenrod | clrBlanchedAlmond |
| clrLemonChiffon  | clrBeige             | clrAntiqueWhite    | clrPapayaWhip       | clrCornsilk      | clrLightYellow  | clrLightCyan      | clrLinen          |
| clrLavender      | clrMistyRose         | clrOldLace         | clrWhiteSmoke       | clrSeashell      | clrIvory        | clrHoneydew       | clrAliceBlue      |
| clrLavenderBlush | clrMintCream         | clrSnow            | clrWhite            |                  |                 |                   |                   |

Die Farbe für Objekte kann durch die Funktion [ObjectSetInteger\(\)](#) und für Benutzerindikatoren durch die Funktion [PlotIndexSetInteger\(\)](#) festgestellt werden. Für Erhalten des Farbwertes dienen Analogfunktionen [ObjectGetInteger\(\)](#) und [PlotIndexGetInteger\(\)](#).

**Beispiel:**

```
//---- indicator settings
#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots 3
#property indicator_type1 DRAW_LINE
#property indicator_type2 DRAW_LINE
#property indicator_type3 DRAW_LINE
#property indicator_color1 clrBlue
#property indicator_color2 clrRed
#property indicator_color3 clrLime
```

## Wingdings

Symbolen der Schrift Wingdings, die mit dem Objekt [OBJ\\_ARROW](#) verwendet werden:

|     |  |     |  |     |  |     |  |     |  |     |  |     |  |     |  |     |  |     |  |     |  |     |  |     |  |     |  |     |  |     |  |
|-----|--|-----|--|-----|--|-----|--|-----|--|-----|--|-----|--|-----|--|-----|--|-----|--|-----|--|-----|--|-----|--|-----|--|-----|--|-----|--|
| 32  |  | 33  |  | 34  |  | 35  |  | 36  |  | 37  |  | 38  |  | 39  |  | 40  |  | 41  |  | 42  |  | 43  |  | 44  |  | 45  |  | 46  |  | 47  |  |
| 48  |  | 49  |  | 50  |  | 51  |  | 52  |  | 53  |  | 54  |  | 55  |  | 56  |  | 57  |  | 58  |  | 59  |  | 60  |  | 61  |  | 62  |  | 63  |  |
| 64  |  | 65  |  | 66  |  | 67  |  | 68  |  | 69  |  | 70  |  | 71  |  | 72  |  | 73  |  | 74  |  | 75  |  | 76  |  | 77  |  | 78  |  | 79  |  |
| 80  |  | 81  |  | 82  |  | 83  |  | 84  |  | 85  |  | 86  |  | 87  |  | 88  |  | 89  |  | 90  |  | 91  |  | 92  |  | 93  |  | 94  |  | 95  |  |
| 96  |  | 97  |  | 98  |  | 99  |  | 100 |  | 101 |  | 102 |  | 103 |  | 104 |  | 105 |  | 106 |  | 107 |  | 108 |  | 109 |  | 110 |  | 111 |  |
| 112 |  | 113 |  | 114 |  | 115 |  | 116 |  | 117 |  | 118 |  | 119 |  | 120 |  | 121 |  | 122 |  | 123 |  | 124 |  | 125 |  | 126 |  | 127 |  |
| 128 |  | 129 |  | 130 |  | 131 |  | 132 |  | 133 |  | 134 |  | 135 |  | 136 |  | 137 |  | 138 |  | 139 |  | 140 |  | 141 |  | 142 |  | 143 |  |
| 144 |  | 145 |  | 146 |  | 147 |  | 148 |  | 149 |  | 150 |  | 151 |  | 152 |  | 153 |  | 154 |  | 155 |  | 156 |  | 157 |  | 158 |  | 159 |  |
| 160 |  | 161 |  | 162 |  | 163 |  | 164 |  | 165 |  | 166 |  | 167 |  | 168 |  | 169 |  | 170 |  | 171 |  | 172 |  | 173 |  | 174 |  | 175 |  |
| 176 |  | 177 |  | 178 |  | 179 |  | 180 |  | 181 |  | 182 |  | 183 |  | 184 |  | 185 |  | 186 |  | 187 |  | 188 |  | 189 |  | 190 |  | 191 |  |
| 192 |  | 193 |  | 194 |  | 195 |  | 196 |  | 197 |  | 198 |  | 199 |  | 200 |  | 201 |  | 202 |  | 203 |  | 204 |  | 205 |  | 206 |  | 207 |  |
| 208 |  | 209 |  | 210 |  | 211 |  | 212 |  | 213 |  | 214 |  | 215 |  | 216 |  | 217 |  | 218 |  | 219 |  | 220 |  | 221 |  | 222 |  | 223 |  |
| 224 |  | 225 |  | 226 |  | 227 |  | 228 |  | 229 |  | 230 |  | 231 |  | 232 |  | 233 |  | 234 |  | 235 |  | 236 |  | 237 |  | 238 |  | 239 |  |
| 240 |  | 241 |  | 242 |  | 243 |  | 244 |  | 245 |  | 246 |  | 247 |  | 248 |  | 249 |  | 250 |  | 251 |  | 252 |  | 253 |  | 254 |  | 255 |  |

Feststellung des notwendigen Symbols erfolgt durch die Funktion [ObjectSetInteger\(\)](#).

Beispiel:

```
void OnStart()
{
    //---
    string up_arrow="up_arrow";
    datetime time=TimeCurrent();
    double lastClose[1];
    int close=CopyClose(Symbol(),Period(),0,1,lastClose);           // wir bekommen Preis
    //--- wenn der Preis erhalten
    if(close>0)
    {
        ObjectCreate(0,up_arrow,OBJ_ARROW,0,0,0,0,0);               // erzeugen wir Weiche
        ObjectSetInteger(0,up_arrow,OBJPROP_ARROWCODE,241);         // stellen wir Weichenkod
        ObjectSetInteger(0,up_arrow,OBJPROP_TIME,time);             // Legen wir Zeit fest
        ObjectSetDouble(0,up_arrow,OBJPROP_PRICE,lastClose[0]);      // legen wir Preis fest
        ChartRedraw(0);                                             // zeichnen wir Fenster r
    }
    else
        Print("Erhalten des letzten Preises Close Fehler! ");
}
```

## Indikatorkonstanten

Vorbestimmt sind 37 [technische Standardindikatoren](#), die in Programmen mit der Sprache MQL5 verwendet werden können. Außerdem kann man seine eigene Benutzerindikatoren durch die Funktion [iCustom\(\)](#) erzeugen. Alle dafür notwendigen Konstanten werden in fünf Gruppen aufgeteilt:

- [Preiskonstanten](#) - für Auswahl des Preis- oder Volumentypes, nach denen Indikator berechnet wird;
- [Glaettungsmethoden](#) - eingebaute Glaettungsmethoden, die in Indikatoren verwendet werden;
- [Linien der Indikatoren](#) - Identifikatoren der Indikatorpuffer beim Zugang zu Werten der Indikatoren durch die Funktion [CopyBuffer\(\)](#);
- [Zeichnungsstile](#) - für Andeutung eines der 18 Typen der Zeichnung und Einstellung des Zeichnungsstils der Linie;
- [Eigenschaften der Benutzindikatoren](#) - werden in Funktionen für Arbeit mit [Benutzerindikatoren](#) verwendet;
- [Typen der Indikatoren](#) - werden für Andeutung des Typs des technischen Indikators bei der Erzeugung von handle durch die Funktion [IndicatorCreate\(\)](#) verwendet;
- [Indikatoren der Datentypen](#) - werden für Festlegung des Datentyps verwendet, die durch das Feld des Typs [MqlParam](#) in die Funktion [IndicatorCreate\(\)](#) übertragen werden.

## Preiskonstanten

Technische Indikatoren brauchen für ihre Berechnungen die Angaben von Preiswerten und/oder Volumenwerten, auf denen sie gerechnet werden. Es gibt 7 vorbestimmte Identifikatoren der Aufzählung `ENUM_APPLIED_PRICE`, für Andeutung der notwendigen Preigrundlage der Berechnungen.

### ENUM\_APPLIED\_PRICE

| Identifikator  | Beschreibung                                                 |
|----------------|--------------------------------------------------------------|
| PRICE_CLOSE    | Eroöffnungspreis                                             |
| PRICE_OPEN     | Schlusspreis                                                 |
| PRICE_HIGH     | Maximaler Preis für die Periode                              |
| PRICE_LOW      | Minimaler Preis für die Periode                              |
| PRICE_MEDIAN   | Medianpreis, $(high+low)/2$                                  |
| PRICE_TYPICAL  | Typischer Preis, $(high+low+close)/3$                        |
| PRICE_WEIGHTED | Durchschnittlich gewogener Preis, $(high+low+close+close)/4$ |

Wenn das Volumen in Berechnungen verwendet wird, muss man einen der zwei Werte der Aufzählung `ENUM_APPLIED_VOLUME` angeben .

### ENUM\_APPLIED\_VOLUME

| Identifikator | Beschreibung   |
|---------------|----------------|
| VOLUME_TICK   | Tickvolumen    |
| VOLUME_REAL   | Handelsvolumen |

Der technische Indikator [`iStochastic\(\)`](#) hat zwei Varianten für seine Berechnung :

- entweder nur die Preise Close;
- oder Preise High und Low.

für Auswahl der notwendigen Variante der Berechnung muss einen der Werte der Aufzählung `ENUM_STO_PRICE` angegeben werden.

### ENUM\_STO\_PRICE

| Identifikator  | Beschreibung                                |
|----------------|---------------------------------------------|
| STO_LOWHIGH    | Berechnung ist nach den Preisen Low/High    |
| STO_CLOSECLOSE | Berechnung ist nach den Preisen Close/Close |

Wenn ein technischer Indikator für seine Berechnungen Preisdaten verwendet, deren Typ durch die Aufzählung `ENUM_APPLIED_PRICE` festgelegt wird, kann als Eingabepreisreihe handle jedes Indikators (der ins Terminal eingebaut wird oder vom Benutzer geschrieben wird) angegeben werden. In diesem Fall werden für Berechnungen Werte des Nullpuffers des Indikators verwendet werden. Das ermöglicht die Werte eines Indikators mittels der Werte eines anderen Indikators leicht zu erzeugen. Handle



eines Benutzerindikators wird durch Aufruf der Funktion [iCustom\(\)](#) erzeugt.

**Beispiel:**

```
#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots 2
//--- input parameters
input int      RSIPeriod=14;          // Periode für die Berechnung von RSI
input int      Smooth=8;              // Glaettungsperiode RSI
input ENUM_MA_METHOD meth=MODE_SMMa; // Glaettungsverfahren
//---- plot RSI
#property indicator_label1  "RSI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//---- plot RSI_Smoothed
#property indicator_label2  "RSI_Smoothed"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrNavy
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- indicator buffers
double      RSIBuffer[];          // hier werden wir die Werte RSI aufbewahren
double      RSI_SmoothedBuffer[]; //hier haben wir Glaettungswerte RSI
int         RSIhandle;           // Handle für Indikator RSI
//+-----+
//| Custom indicator initialization function |
//+-----+
void OnInit()
{
//--- indicator buffers mapping
    SetIndexBuffer(0,RSIBuffer,INDICATOR_DATA);
    SetIndexBuffer(1,RSI_SmoothedBuffer,INDICATOR_DATA);
    IndicatorSetString(INDICATOR_SHORTNAME,"iRSI");
    IndicatorSetInteger(INDICATOR_DIGITS,2);
//---
    RSIhandle=iRSI(NULL,0,RSIPeriod,PRICE_CLOSE);
//---
; }
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const int begin,
               const double &price[]
               )
```

```

{
//--- stellen wir Wert des letzten Fehlers auf Null
    ResetLastError();
//--- bekommen wir Daten des Indikators RSI im Feld RSIBuffer[]
    int copied=CopyBuffer(RSIhandle,0,0,rates_total,RSIBuffer);
    if(copied<=0)
    {
        Print("Kopieren des Indikatorwertes Fehler RSI. Error = ",
              GetLastError()," ", copied ="",copied);
        return(0);
    }
//--- using values of RSI erzeugen wir Indikator der Mittelwerte mit Verwendung der F
    int RSI_MA_handle=iMA(NULL,0,Smooth,0,method,RSIhandle);
    copied=CopyBuffer(RSI_MA_handle,0,0,rates_total,RSI_SmoothedBuffer);
    if(copied<=0)
    {
        Print("Kopieren des gerichteten Feldes Fehler RSI. Error = ",
              GetLastError()," ", copied ="",copied);
        return(0);
    }
//--- return value of prev_calculated for next call
    return(rates_total);
}

```

## Glaettungsmethoden

Vielen technischen Indikatoren zugrunde liegt eine oder andere Methode der Mittelung von Preissereien. Einige technische Standardindikatoren fordern der Andeutung des Mittelungstyps als Parameter. Der Andeutung des notwendigen Typs der Mittelung dienen Identifikatoren, die in der Aufzählung ENUM\_MA\_METHOD angegeben sind .

### ENUM\_MA\_METHOD

| Identifikator | Beschreibung                 |
|---------------|------------------------------|
| MODE_SMA      | einfache Mittelung           |
| MODE_EMA      | exponentielle Mittelung      |
| MODE_SMMA     | geglättete Mittelung         |
| MODE_LWMA     | lineal ausgewogene Mittelung |

### Beispiel:

```
double ExtJaws[];
double ExtTeeth[];
double ExtLips[];
//---- handles for moving averages
int    ExtJawsHandle;
int    ExtTeethHandle;
int    ExtLipsHandle;
//--- get MA's handles
ExtJawsHandle=iMA(NULL,0,JawsPeriod,0,MODE_SMMA,PRICE_MEDIAN);
ExtTeethHandle=iMA(NULL,0,TeethPeriod,0,MODE_SMMA,PRICE_MEDIAN);
ExtLipsHandle=iMA(NULL,0,LipsPeriod,0,MODE_SMMA,PRICE_MEDIAN);
```

## Linien der Indikatoren

Einige [technische Indikatoren](#) haben mehrere auf dem Chart gezeichnete Puffer. Numerierung der Indikatorpuffers faengt mit 0 an. Beim Kopieren der Indikatorwerte durch die Funktion [CopyBuffer\(\)](#) ins Feld des Typs double kann man für einige Indikatoren nicht die Nummer des kopierten Puffers, sondern den Identifikator dieses Puffers angeben.

Identifikatoren der Indikatorlinien, zulaessige für Kopieren der Indikatorwerte [iMACD\(\)](#), [iRVI\(\)](#) und [iStochastic\(\)](#)

| Konstante   | Wert | Beschreibung |
|-------------|------|--------------|
| MAIN_LINE   | 0    | Grundlinie   |
| SIGNAL_LINE | 1    | Sygnallinie  |

Identifikatoren der Indikatorlinien, die beim Kopieren der Indikatorwerte [ADX\(\)](#) und [ADXW\(\)](#) zulaessig sind

| Konstante    | Wert | Beschreibung |
|--------------|------|--------------|
| MAIN_LINE    | 0    | Grundlinie   |
| PLUSDI_LINE  | 1    | Linie +DI    |
| MINUSDI_LINE | 2    | Linie -DI    |

Identifikatoren der Indikatorlinien, die beim Kopieren der Indikatorwerte [iBands\(\)](#) zulaessig sind

| Konstante  | Wert | Beschreibung  |
|------------|------|---------------|
| BASE_LINE  | 0    | Grundlinie    |
| UPPER_BAND | 1    | Obere Grenze  |
| LOWER_BAND | 2    | Untere Grenze |

Identifikatoren der Indikatorlinien, die beim Kopieren der Indikatorwerte [iEnvelopes\(\)](#) und [iFractals\(\)](#) zulaessig sind

| Konstante  | Wert | Beschreibung |
|------------|------|--------------|
| UPPER_LINE | 0    | Obere Linie  |
| LOWER_LINE | 1    | Untere Linie |

Identifikatoren der Indikatorlinien, die beim Kopieren der Werte des Indikators [iGator\(\)](#) zulaessig sind

| Konstante       | Wert | Beschreibung       |
|-----------------|------|--------------------|
| UPPER_HISTOGRAM | 0    | Oberes Histogramm  |
| LOWER_HISTOGRAM | 2    | Unteres Histogramm |

Identifikatoren der Indikatorlinien, die beim Kopieren der Werte des Indikators [iAlligator\(\)](#) zulaessig sind

| Konstante       | Wert | Beschreibung     |
|-----------------|------|------------------|
| GATORJAW_LINE   | 0    | Linie der Kiefer |
| GATORTEETH_LINE | 1    | Linie der Zaehne |
| GATORLIPS_LINE  | 2    | Linie der Lippen |

Identifikatoren der Indikatorlinien, die beim Kopieren der Werte des Indikators [ilchimoku\(\)](#) zulaessig sind

| Konstante        | Wert | Beschreibung        |
|------------------|------|---------------------|
| TENKANSEN_LINE   | 0    | Linie Tenkan-sen    |
| KIJUNSEN_LINE    | 1    | Linie Kijun-sen     |
| SENKOSPAN_A_LINE | 2    | Linie Senkou Span A |
| SENKOSPAN_B_LINE | 3    | Linie Senkou Span B |
| CHINKOSPAN_LINE  | 4    | Linie Chinkou Span  |

## Zeichnungsstile

Bei Erzeugung der [Benutzerindikatoren](#) kann einer der 18 Typen der graphischen Konstruktion (Darstellungsart im Hauptfenster des Charts oder im Subfenster des Charts) angegeben werden, dessen Werte in der Aufzählung `ENUM_DRAW_TYPE` spezifiziert werden.

Es ist zulaessig, in einem Benutzerindikator beliebige Arten der [Konstruktion/Zeichnung der Indikatoren](#) zu verwenden. Für jeden Konstruktionstyp muss von 1 bis 5 [globale Felder](#) für Datenspeicherung angegeben werden, die für die Zeichnung erforderlich sind. Diese Datenfelder muessen mit Indikatorpuffern durch die Funktion [SetIndexBuffer\(\)](#) verbunden werden, und für jeden Puffer muss der Datentyp von der Aufzählung [ENUM\\_INDEXBUFFER\\_TYPE](#) angegeben werden.

Abhängig vom Zeichnungsstil können Sie von 1 bis 4 Wertpuffer brauchen (vermerkt als `INDICATOR_DATA`), wenn der dynamische Farbenwechsel vom Stil zugelaest wird (aale Stile mit dem Wort `COLOR` im Namen), braucht man auch einen Farbenpuffer (angegeben Typ `INDICATOR_COLOR_INDEX`). Farbenpuffer wird immer gebunden nach den dem Stil entsprechenden Wertpuffern.

### ENUM\_DRAW\_TYPE

| Identifikator                        | Beschreibung                               | Werte der Puffer | Farbe der Puffer |
|--------------------------------------|--------------------------------------------|------------------|------------------|
| <a href="#">DRAW_NONE</a>            | Nicht gezeichnet                           | 1                | 0                |
| <a href="#">DRAW_LINE</a>            | Linie                                      | 1                | 0                |
| <a href="#">DRAW_SECTION</a>         | Abschnitte                                 | 1                | 0                |
| <a href="#">DRAW_HISTOGRAM</a>       | Histogramm von der Nulllinie               | 1                | 0                |
| <a href="#">DRAW_HISTOGRAM2</a>      | Histogramm der 2 Indikatorpuffer           | 2                | 0                |
| <a href="#">DRAW_ARROW</a>           | Weichenaufzeichnung                        | 1                | 0                |
| <a href="#">DRAW_ZIGZAG</a>          | Stil Zigzag laesst vertikale Abschnitte zu | 2                | 0                |
| <a href="#">DRAW_FILLING</a>         | Farbenfuellung zwischen zwei Levels        | 2                | 0                |
| <a href="#">DRAW_BARS</a>            | Darstellung als Folge von Bars             | 4                | 0                |
| <a href="#">DRAW_CANDLES</a>         | Darstellung als Folge von Kerzen           | 4                | 0                |
| <a href="#">DRAW_COLOR_LINE</a>      | Mehrfarbige Linie                          | 1                | 1                |
| <a href="#">DRAW_COLOR_SECTION</a>   | Mehrfarbige Abschnitte                     | 1                | 1                |
| <a href="#">DRAW_COLOR_HISTOGRAM</a> | Mehrfarbiges Histogramm von der Nulllinie  | 1                | 1                |

|                                              |                                                  |   |   |
|----------------------------------------------|--------------------------------------------------|---|---|
| <a href="#"><u>DRAW_COLOR_HISTOGRAM2</u></a> | Mehrfarbiges Histogramm der zwei Indikatorpuffer | 2 | 1 |
| <a href="#"><u>DRAW_COLOR_ARROW</u></a>      | Zeichnung mit mehrfarbigen Weichen               | 1 | 1 |
| <a href="#"><u>DRAW_COLOR_ZIGZAG</u></a>     | Mehrfarbiger ZigZag                              | 2 | 1 |
| <a href="#"><u>DRAW_COLOR_BARS</u></a>       | Mehrfarbige Bars                                 | 4 | 1 |
| <a href="#"><u>DRAW_COLOR_CANDLE</u></a>     | Mehrfarbige Kerzen                               | 4 | 1 |

für Praezisierung der Darstellung der gewaehlten Zeichnungsart werden Identifikatoren verwendet, die in Aufzählungen ENUM\_PLOT\_PROPERTY aufgezählt werden.

für Funktionen [PlotIndexSetInteger\(\)](#) und [PlotIndexGetInteger\(\)](#)

#### ENUM\_PLOT\_PROPERTY\_INTEGER

| Identifikator      | Beschreibung                                                                | Typ der Eigenschaft                     |
|--------------------|-----------------------------------------------------------------------------|-----------------------------------------|
| PLOT_ARROW         | Weichenkode für Stil DRAW_ARROW                                             | uchar                                   |
| PLOT_ARROW_SHIFT   | Vertikale Verschiebung der Weichen für Stil DRAW_ARROW                      | int                                     |
| PLOT_DRAW_BEGIN    | Zahl der Anfangsbars ohne Zeichnung und Werte in DataWindow                 | int                                     |
| PLOT_DRAW_TYPE     | Typ der graphischen Konstruktion                                            | <a href="#"><u>ENUM_DRAW_TYPE</u></a>   |
| PLOT_SHOW_DATA     | Merkmal der Darstellung von Konstruktionswerte im Fenster DataWindow        | bool                                    |
| PLOT_SHIFT         | Verschiebung der graphischen Konstruktion des Indikators Zeit-Achse in Bars | int                                     |
| PLOT_LINE_STYLE    | Stil der zeichnerischen Linie                                               | <a href="#"><u>ENUM_LINE_STYLE</u></a>  |
| PLOT_LINE_WIDTH    | Breite der zeichnerischen Linie                                             | int                                     |
| PLOT_COLOR_INDEXES | Anzahl der Farben                                                           | int                                     |
| PLOT_LINE_COLOR    | Pufferindex mit zeichnerischer Farbe                                        | color Modifikator=Nummer der Indexfarbe |

für die Funktion [PlotIndexSetDouble\(\)](#)

#### ENUM\_PLOT\_PROPERTY\_DOUBLE

| Identifikator    | Beschreibung                                               | Typ der Eigenschaft |
|------------------|------------------------------------------------------------|---------------------|
| PLOT_EMPTY_VALUE | Leerwert für Konstruktion, für die es keine Zeichnung gibt | double              |

für die Funktion [PlotIndexSetString\(\)](#)

#### ENUM\_PLOT\_PROPERTY\_STRING

| Identifikator | Beschreibung                                                                                                                                                                                                                                                                                   | Typ der Eigenschaft |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------|
| PLOT_LABEL    | Name der graphischen Indikatorserie für Darstellung im Fenster DataWindow. Für komplexe grafische Stile, die mehrere Indikatorpuffer für Anzeige brauchen, können die Namen von jedem Puffer mit Trennzeichen ";" angegeben werden. Beispiel-Code wird in <a href="#">DRAW_CANDLES</a> gezeigt | string              |

Es gibt 5 Stile mit denen die Linie im Benutzerindikator gezeichnet werden kann. Sie sind nur bei der Linienbreite 0 oder 1 anwendbar.

#### ENUM\_LINE\_STYLE

| Identifikator    | Beschreibung             |
|------------------|--------------------------|
| STYLE_SOLID      | Ausgezogene Linie        |
| STYLE_DASH       | diskontinuierliche Linie |
| STYLE_DOT        | Punktierte Linie         |
| STYLE_DASHDOT    | Strichpunktlinie         |
| STYLE_DASHDOTDOT | Strich-zwei Punkte       |

für Einstellung des Stils der Liniezeichnung und Zeichnungsart wird die Funktion [PlotIndexSetInteger\(\)](#) verwendet. Für Fibonacci-Erweiterungen können die Breite und Stil der Zeichnung durch die Funktion [ObjectSetInteger\(\)](#) angegeben werden.

#### Beispiel:

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots 1
//--- indicator buffers
double      MABuffer[];
```



```

//+-----+
//| Custom indicator initialization function |
//+-----+
void OnInit()
{
//--- Bindung des Feldes zum Indikatorpuffer mit Index 0
    SetIndexBuffer(0,MABuffer,INDICATOR_DATA);
//--- Einstellung der Liniezeichnung
    PlotIndexSetInteger(0,PLOT_DRAW_TYPE,DRAW_LINE);
//--- Einstellung des Stils für Liniezeichnung
    PlotIndexSetInteger(0,PLOT_LINE_STYLE,STYLE_DOT);
//--- Einstellung der Linienfarbe
    PlotIndexSetInteger(0,PLOT_LINE_COLOR,clrRed);
//--- Einstellung der Linienbreite
    PlotIndexSetInteger(0,PLOT_LINE_WIDTH,1);
//--- Einstellung der Marke für die Linie
    PlotIndexSetString(0,PLOT_LABEL,"Moving Average");
//---
; }
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
//---
    for(int i=prev_calculated;i<rates_total;i++)
    {
        MABuffer[i]=close[i];
    }
//--- return value of prev_calculated for next call
    return(rates_total);
; }

```

## Eigenschaften der Benutzeranzeiger

Zahl der Indikatorpuffer, die im Benutzerindikator verwendet werden können ist unbegrenzt. Aber es ist notwendig, für jedes Feld, das als Indikatorpuffer durch die Funktion [SetIndexBuffer\(\)](#) zugeordnet wird, muss der Datentyp angegeben werden, den es aufbewahren wird. Das kann einer der Werte der Aufzählung ENUM\_INDEXBUFFER\_TYPE sein.

### ENUM\_INDEXBUFFER\_TYPE

| Identifikator          | Beschreibung                         |
|------------------------|--------------------------------------|
| INDICATOR_DATA         | Angaben für Zeichnen                 |
| INDICATOR_COLOR_INDEX  | Farbe                                |
| INDICATOR_CALCULATIONS | Hilfspuffer für Zwischenberechnungen |

Benutzerindikator hat viele Einstellungen für Darstellung und Erfassen. Diese Einstellungen werden durch Festlegung der entsprechenden Funktionen des Indikators durch die Funktionen [IndicatorSetDouble\(\)](#), [IndicatorSetInteger\(\)](#) und [IndicatorSetString\(\)](#) durchgeführt. Identifikatoren der Indikatoreigenschaften gibt es in Aufzählungen ENUM\_CUSTOMIND\_PROPERTY.

### ENUM\_CUSTOMIND\_PROPERTY\_INTEGER

| Identifikator        | Beschreibung                                                                                                    | Typ der Eigenschaft                                        |
|----------------------|-----------------------------------------------------------------------------------------------------------------|------------------------------------------------------------|
| INDICATOR_DIGITS     | Genauigkeit der Darstellung von Indikatorwerten                                                                 | int                                                        |
| INDICATOR_HEIGHT     | Feste Höhe von Indikators eigenen Fenster (Befehl von Preprozessor <a href="#">#property indicator_height</a> ) | int                                                        |
| INDICATOR_LEVELS     | Anzahl der Levels im Indikatorfenster                                                                           | int                                                        |
| INDICATOR_LEVELCOLOR | Farbe der Levellinie                                                                                            | color<br>Modifikator=Levelnummer                           |
| INDICATOR_LEVELSTYLE | Stil der Levellinie                                                                                             | <a href="#">ENUM_LINE_STYLE</a><br>Modifiaktor=Levelnummer |
| INDICATOR_LEVELWIDTH | Breite der Levellinie                                                                                           | int<br>Modifikator=Levelnummer                             |

### ENUM\_CUSTOMIND\_PROPERTY\_DOUBLE

| Identifikator     | Beschreibung                  | Typ der Eigenschaft |
|-------------------|-------------------------------|---------------------|
| INDICATOR_MINIMUM | Minimum des Indikatorfensters | double              |
| INDICATOR_MAXIMUM | Maximum des Indikatorfensters | double              |

|                      |           |                                   |
|----------------------|-----------|-----------------------------------|
| INDICATOR_LEVELVALUE | Levelwert | double<br>Modifikator=Levelnummer |
|----------------------|-----------|-----------------------------------|

ENUM\_CUSTOMIND\_PROPERTY\_STRING

| Identifikator       | Beschreibung         | Typ der Eigenschaft               |
|---------------------|----------------------|-----------------------------------|
| INDICATOR_SHORTNAME | Kurzer Indikatorname | string                            |
| INDICATOR_LEVELTEXT | Levelbeschreibung    | string<br>Modifikator=Levelnummer |

Beispiele:

```
//--- indicator settings
#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots 2
#property indicator_type1 DRAW_LINE
#property indicator_type2 DRAW_LINE
#property indicator_color1 clrLightSeaGreen
#property indicator_color2 clrRed
//--- input parameters
extern int KPeriod=5;
extern int DPeriod=3;
extern int Slowing=3;
//--- indicator buffers
double MainBuffer[];
double SignalBuffer[];
double HighesBuffer[];
double LowesBuffer[];

//+-----+
//| Custom indicator initialization function |
//+-----+

void OnInit()
{
//--- indicator buffers mapping
SetIndexBuffer(0,MainBuffer,INDICATOR_DATA);
SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
SetIndexBuffer(2,HighesBuffer,INDICATOR_CALCULATIONS);
SetIndexBuffer(3,LowesBuffer,INDICATOR_CALCULATIONS);
//--- set accuracy
IndicatorSetInteger(INDICATOR_DIGITS,2);
//--- set levels
IndicatorSetInteger(INDICATOR_LEVELS,2);
IndicatorSetDouble(INDICATOR_LEVELVALUE,0,20);
IndicatorSetDouble(INDICATOR_LEVELVALUE,1,80);
//--- set maximum and minimum for subwindow
IndicatorSetDouble(INDICATOR_MINIMUM,0);
IndicatorSetDouble(INDICATOR_MAXIMUM,100);
```

```
//--- sets first bar from what index will be drawn
PlotIndexSetInteger(0,PLOT_DRAW_BEGIN,KPeriod+Slowing-2);
PlotIndexSetInteger(1,PLOT_DRAW_BEGIN,KPeriod+Slowing+DPeriod);
//--- set style STYLE_DOT for second line
PlotIndexSetInteger(1,PLOT_LINE_STYLE,STYLE_DOT);
//--- name for DataWindow and indicator subwindow label
IndicatorSetString(INDICATOR_SHORTNAME,"Stoch("+KPeriod+", "+DPeriod+", "+Slowing+")'
PlotIndexSetString(0,PLOT_LABEL,"Main");
PlotIndexSetString(1,PLOT_LABEL,"Signal");
//--- sets drawing line empty value
PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0.0);
PlotIndexSetDouble(1,PLOT_EMPTY_VALUE,0.0);
//--- initialization done
;}
```

## Typen der technischen Indikatoren

Es gibt 2 Wege, handle eines Indikators für weiteren [Zugang zu seinen Elementen](#) programmgesteuert zu erzeugen. Der erste Weg ist die direkte Andeutung des Funktionsnamens aus der Liste der [technischen Indikatoren](#). Der zweite Weg ermöglicht durch die Funktion [IndicatorCreate\(\)](#) handle jedes Indikators durch Zuordnung des Identifikators aus der Aufzählung `ENUM_INDICATOR` einheitlich zu erzeugen. Beide Varianten der Erzeugung von handle sind gleichgestellt, Sie können diejenige verwenden, die bei der Schreibung des MQL5 Programms für Sie am meisten passt.

Bei der Erzeugung eines Indikators des Typs `IND_CUSTOM`, muss das Feld *type* des ersten Elements des Feldes der [Eingabeparameter MqlParam](#) den Wert `TYPE_STRING` aus der Aufzählung [ENUM\\_DATATYPE](#) haben, und das Feld *string\_value* des ersten Elements den Namen des Benutzerindikators enthalten.

### ENUM\_INDICATOR

| Identifikator | Indikator                         |
|---------------|-----------------------------------|
| IND_AC        | Accelerator Oscillator            |
| IND_AD        | Accumulation/Distribution         |
| IND_ADX       | Average Directional Index         |
| IND_ADXW      | ADX by Welles Wilder              |
| IND_ALLIGATOR | Alligator                         |
| IND_AMA       | Adaptive Moving Average           |
| IND_AO        | Awesome Oscillator                |
| IND_ATR       | Average True Range                |
| IND_BANDS     | Bollinger Bands®                  |
| IND_BEARS     | Bears Power                       |
| IND_BULLS     | Bulls Power                       |
| IND_BWMFI     | Market Facilitation Index         |
| IND_CCI       | Commodity Channel Index           |
| IND_CHAIKIN   | Chaikin Oscillator                |
| IND_CUSTOM    | Custom indicator                  |
| IND_DEMA      | Double Exponential Moving Average |
| IND_DEMARKER  | DeMarker                          |
| IND_ENVELOPES | Envelopes                         |
| IND_FORCE     | Force Index                       |
| IND_FRACTALS  | Fractals                          |
| IND_FRAMA     | Fractal Adaptive Moving Average   |
| IND_GATOR     | Gator Oscillator                  |

|                |                                               |
|----------------|-----------------------------------------------|
| IND_ICHIMOKU   | Ichimoku Kinko Hyo                            |
| IND_MA         | Moving Average                                |
| IND_MACD       | MACD                                          |
| IND_MFI        | Money Flow Index                              |
| IND_MOMENTUM   | Momentum                                      |
| IND_OBV        | On Balance Volume                             |
| IND_OSMA       | OsMA                                          |
| IND_RSI        | Relative Strength Index                       |
| IND_RVI        | Relative Vigor Index                          |
| IND_SAR        | Parabolic SAR                                 |
| IND_STDDEV     | Standard Deviation                            |
| IND_STOCHASTIC | Stochastic Oscillator                         |
| IND_TEMA       | Triple Exponential Moving Average             |
| IND_TRIX       | Triple Exponential Moving Averages Oscillator |
| IND_VIDYA      | Variable Index Dynamic Average                |
| IND_VOLUMES    | Volumes                                       |
| IND_WPR        | Williams' Percent Range                       |

## Identifikatoren der Datentypen

Bei der Erzeugung von handle des Indikators durch die Funktion [IndicatorCreate\(\)](#) muss man das Feld des Typs [MqlParam](#) als letzter Parameter angeben. So hat die Struktur [MqlParam](#), die Indikatorparameter beschreibt, ein spezielles Feld *type*. In diesem Feld gibt es Information über Datentyp ([Realtyp](#), [ganzzahliger\\_Typ](#) oder [Zeilentyp](#)), die vom entsprechenden Element dieses Feldes übertragen werden. Wert dieses Feldes der Struktur [MqlParam](#) kann einer der Aufzählungswerte `ENUM_DATATYPE` sein.

### ENUM\_DATATYPE

| Identifikator | Datentyp |
|---------------|----------|
| TYPE_BOOL     | bool     |
| TYPE_CHAR     | char     |
| TYPE_UCHAR    | uchar    |
| TYPE_SHORT    | short    |
| TYPE_USHORT   | ushort   |
| TYPE_COLOR    | color    |
| TYPE_INT      | int      |
| TYPE_UINT     | uint     |
| TYPE_DATETIME | datetime |
| TYPE_LONG     | long     |
| TYPE_ULONG    | ulong    |
| TYPE_FLOAT    | float    |
| TYPE_DOUBLE   | double   |
| TYPE_STRING   | string   |

Jedes Element dieses Feldes beschreibt den entsprechenden Eingabeparameter des erzeugten [technischen Indikators](#), darum müssen der Typ und Reihenfolge der Elemente des Feldes entsprechend der Beschreibung strikt gehalten werden.

## Medium Zustand

Konstanten, die laufendes Medium von mql5-Programms beschreiben, werden in Gruppen aufgeteilt:

- [Kundenterminalstatus](#) - Information über das Kundenterminal;
- [Information über das abgelaufene MQL5-Programm](#) -Eigenschaften des mql5-Programms, die sein Verhalten zu steuern helfen;
- [Symboleigenschaften](#) - Erhalten von Handelsinformation über das Symbol;
- [Kontoeigenschaften](#) - Information über das laufende Handelskonto;
- [Teststatistik](#) - Testergebnisse eine Expert Advisors.



## Zustand des Kundenterminals

Identifikatoren für Erhaltung der Information über das Kundenterminal durch die Funktionen [TerminalInfoInteger\(\)](#) und [TerminalInfoString\(\)](#). Als Parameter nehmen diese Funktionen Werte aus den Aufzählungen ENUM\_TERMINAL\_INFO\_INTEGER und ENUM\_TERMINAL\_INFO\_STRING an.

### ENUM\_TERMINAL\_INFO\_INTEGER

| Identifikator                  | Beschreibung                                                                                                                  | Typ der Eigenschaft |
|--------------------------------|-------------------------------------------------------------------------------------------------------------------------------|---------------------|
| TERMINAL_BUILD                 | Buildnummer des abgelaufenen Terminals                                                                                        | int                 |
| TERMINAL_COMMUNITY_ACCOUNT     | Flagge der Verfügbarkeit von MQL5.community-Autorisierungsdaten im Terminal                                                   | bool                |
| TERMINAL_COMMUNITY_CONNECTION  | Verbindung zum MQL5.community                                                                                                 | bool                |
| TERMINAL_CONNECTED             | Verbindung mit dem Handelsserver                                                                                              | bool                |
| TERMINAL_DLLS_ALLOWED          | Erlaubnis, DLL zu verwenden                                                                                                   | bool                |
| TERMINAL_TRADE_ALLOWED         | <a href="#">Erlaubnis zu handeln</a>                                                                                          | bool                |
| TERMINAL_EMAIL_ENABLED         | Erlaubnis, Briefe zu senden mit der Verwendung von SMTP-Server und Login, die in Einstellungen des Terminals angegeben werden | bool                |
| TERMINAL_FTP_ENABLED           | Erlaubnis, Berichte via FTP zu senden an das angegebene Server für das in Terminaleinstellungen angegebene Handelskonto       | bool                |
| TERMINAL_NOTIFICATIONS_ENABLED | Erlaubnis die Benachrichtigungen auf Smartphone zu senden                                                                     | bool                |
| TERMINAL_MAXBARS               | Maximale Anzahl der Bars auf dem Chart                                                                                        | int                 |
| TERMINAL_MQID                  | Flagge der Verfügbarkeit von MetaQuotes ID um <a href="#">Push-Benachrichtigungen</a> zu senden                               | bool                |
| TERMINAL_CODEPAGE              | Nummer <a href="#">der Kodeseite der Sprache</a> , die im Kundenterminal installiert wird                                     | int                 |
| TERMINAL_CPU_CORES             | Die Anzahl der CPU-Kerne im                                                                                                   | int                 |

|                           | System                                                                                                              |      |
|---------------------------|---------------------------------------------------------------------------------------------------------------------|------|
| TERMINAL_DISK_SPACE       | Freier Festplattenspeicher für den Ordner MQL5\Files des Terminals (des Agenten), in MB                             | int  |
| TERMINAL_MEMORY_PHYSICAL  | Physikalischer Speicher im System, in Mb                                                                            | int  |
| TERMINAL_MEMORY_TOTAL     | Speicher der dem Verfahren des Terminals (des Agenten) verfügbar ist, in MB                                         | int  |
| TERMINAL_MEMORY_AVAILABLE | Freier Speicher vom verfahren des Terminals (den Agenten), in MB                                                    | int  |
| TERMINAL_MEMORY_USED      | Die Größe des durch das Terminal (den Agent) verwendeten Speichers, in MB                                           | int  |
| TERMINAL_X64              | Die Anzeige von "64-Bit-Terminal"                                                                                   | bool |
| TERMINAL_OPENCL_SUPPORT   | Die version von unterstützten OpenCL in Form von 0x00010002 = 1.2. "0" bedeutet, dass OpenCL nicht unterstützt wird | int  |

## ENUM\_TERMINAL\_INFO\_DOUBLE

| Bezeichner                 | Beschreibung                             | Typ der Eigenschaft |
|----------------------------|------------------------------------------|---------------------|
| TERMINAL_COMMUNITY_BALANCE | Balance des Benutzers auf MQL5.community | double              |

[Dateioperationen](#) können nur in zwei Verzeichnissen durchgeführt werden, Pfade zu denen durch die Anforderung der Eigenschaften TERMINAL\_DATA\_PATH und TERMINAL\_COMMONDATA\_PATH erhalten werden können.

## ENUM\_TERMINAL\_INFO\_STRING

| Identifikator     | Beschreibung            | Typ der Eigenschaft |
|-------------------|-------------------------|---------------------|
| TERMINAL_LANGUAGE | Sprache des Terminals   | string              |
| TERMINAL_COMPANY  | Name der Gesellschaft   | string              |
| TERMINAL_NAME     | Terminalname            | string              |
| TERMINAL_PATH     | Ordner aus dem Terminal | string              |

|                          |                                                                        |        |
|--------------------------|------------------------------------------------------------------------|--------|
|                          | gestartet wird                                                         |        |
| TERMINAL_DATA_PATH       | Ordner, wo Terminladaten aufbewahren werden                            | string |
| TERMINAL_COMMONDATA_PATH | Gesamtordner aller Kundenterminals, die im Computer installiert werden | string |

für bessere Verstaendigung der Pfade, die in Eigenschaften der Parameter `TERMINAL_PATH`, `TERMINAL_DATA_PATH` und `TERMINAL_COMMONDATA_PATH` aufbewahren werden, ist es empfehlenswert, Script durchzuführen, das gibt diese Werte für die Kopie des Terminal, das in Ihrem Computer installiert wird, zurück.

#### Beispiel: Script gibt die Information über Terminalpfade zurück

```
//+-----+
//|                                     Check_TerminalPaths.mq5 |
//|                                     Copyright 2009, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//---
    Print("TERMINAL_PATH = ",TerminalInfoString(TERMINAL_PATH));
    Print("TERMINAL_DATA_PATH = ",TerminalInfoString(TERMINAL_DATA_PATH));
    Print("TERMINAL_COMMONDATA_PATH = ",TerminalInfoString(TERMINAL_COMMONDATA_PATH));
};
```

Als Ergebnis der Durchführung von Script können Sie in the Experts Journal Meldungen sehen, wie unten angegeben werden.

| Message                                                                                                |
|--------------------------------------------------------------------------------------------------------|
| TERMINAL_COMMONDATA_PATH = C:\Documents and Settings\All Users\Application Data\MetaQuotes\Terminal... |
| TERMINAL_DATA_PATH = C:\Program Files\MetaTrader 5                                                     |
| TERMINAL_PATH = C:\Program Files\MetaTrader 5                                                          |

## Information über das ausgeführte MQL5-Programm

Konstanten für Erhaltung der Information über mql5-Programm, werden in `ENUM_MQL_INFO_INTEGER` und `ENUM_MQL_INFO_STRING` aufgezählt.

für die Funktion [MQLInfoInteger\(\)](#)

### ENUM\_MQL\_INFO\_INTEGER

| Identifikator                    | Beschreibung                                                                                                 | Typ der Eigenschaft               |
|----------------------------------|--------------------------------------------------------------------------------------------------------------|-----------------------------------|
| <code>MQL_MEMORY_LIMIT</code>    | Die maximale Größe der dynamischen Speicher für eine MQL5-Programme in MB                                    | int                               |
| <code>MQL_MEMORY_USED</code>     | Die Größe der von MQL5-Programme verwendeten Speicher in MB                                                  | int                               |
| <code>MQL_PROGRAM_TYPE</code>    | Typ des MQL5-Programms                                                                                       | <a href="#">ENUM_PROGRAM_TYPE</a> |
| <code>MQL_DLLS_ALLOWED</code>    | Erlaubnis, DLL für das <b>vorgegebene ausgeführte Programm zu verwenden</b>                                  | bool                              |
| <code>MQL_TRADE_ALLOWED</code>   | <a href="#">Erlaubnis für das vorgegebene ausgeführte Programm zu handeln</a>                                | bool                              |
| <code>MQL_SIGNALS_ALLOWED</code> | Erlaubnis für <b>dieses laufenden Programm mit Signale zu arbeiten</b>                                       | bool                              |
| <code>MQL_DEBUG</code>           | Flagge, das Debugging Mode des Programms angibt                                                              | bool                              |
| <code>MQL_PROFILER</code>        | Flagge der Arbeit des ausgeführten Programms im Modus von Code Profiling                                     | bool                              |
| <code>MQL_TESTER</code>          | Flagge der Arbeit des ausgeführten Programms im Tester                                                       | bool                              |
| <code>MQL_OPTIMIZATION</code>    | Flagge der Arbeit des ausgeführten Programms im Prozess der Optimierung                                      | bool                              |
| <code>MQL_VISUAL_MODE</code>     | Flagge der Arbeit des ausgeführten Programms im visuellen Testenmode                                         | bool                              |
| <code>MQL_FRAME_MODE</code>      | Zeichen, dass der auf dem Chart laufende Experte in <a href="#">Frames von Optimierung-Egebnisse sammelt</a> | bool                              |

|                  |                                                                                                                                                    |                                   |
|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------|
| MQL_LICENSE_TYPE | Art der Lizenz des EX5-Moduls. Die Lizenz bezieht sich auf den EX5-Modul, aus denen ein Antrag mit MQLInfoInteger (MQL_LICENSE_TYPE) gestellt wird | <a href="#">ENUM_LICENSE_TYPE</a> |
|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------|

für die Funktion [MQLInfoString\(\)](#)

#### ENUM\_MQL\_INFO\_STRING

| Identifikator    | Beschreibung                                         | Typ der Eigenschaft |
|------------------|------------------------------------------------------|---------------------|
| MQL_PROGRAM_NAME | Name des ausgeführten MQL5-Programms                 | string              |
| MQL_PROGRAM_PATH | Pfad für das <b>vorgegebene ausgeführte Programm</b> | string              |

für Erhaltung der Information über den Typ des ausgeführten Programms sind Werte der Aufzählung ENUM\_PROGRAM\_TYPE bestimmt.

#### ENUM\_PROGRAM\_TYPE

| Identifikator     | Beschreibung |
|-------------------|--------------|
| PROGRAM_SCRIPT    | Skript       |
| PROGRAM_EXPERT    | Expert       |
| PROGRAM_INDICATOR | Anzeiger     |

#### ENUM\_LICENSE\_TYPE

| Identifikator | Beschreibung                                                                                                                                |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| LICENSE_FREE  | Eine Kostenlose Vollversion                                                                                                                 |
| LICENSE_DEMO  | Eine Testversion von einem bezahlten Produkt vom Markt. Es funktioniert nur im Strategie-Tester                                             |
| LICENSE_FULL  | Eine gekaufte lizenzierte Version erlaubt mindestens fünf Aktivierungen. Der Verkäufer kann die Anzahl der zulässigen Aktivierungen erhöhen |
| LICENSE_TIME  | Eine Version mit der zeitlich begrenzten Lizenz                                                                                             |

Beispiel:

```
ENUM_PROGRAM_TYPE mql_program=(ENUM_PROGRAM_TYPE)MQLInfoInteger(MQL_PROGRAM_TYPE);
switch(mql_program)
{
    case PROGRAM_SCRIPT:
    {
        Print(__FILE__+" is script");
        break;
    }
    case PROGRAM_EXPERT:
    {
        Print(__FILE__+" is Expert Advisor");
        break;
    }
    case PROGRAM_INDICATOR:
    {
        Print(__FILE__+" is custom indicator");
        break;
    }
    default:Print("MQL5 program type value is ",mql_program);
}
```

## Information über das Symbol

für die Erhaltung der laufenden Marktinformation werden Funktionen [SymbolInfoInteger\(\)](#), [SymbolInfoDouble\(\)](#) und [SymbolInfoString\(\)](#) verwendet. Als zweiter Parameter dieser Funktionen ist es zulässig, einen der Identifikatoren von den Aufzählungen `ENUM_SYMBOL_INFO_INTEGER`, `ENUM_SYMBOL_INFO_DOUBLE` und `ENUM_SYMBOL_INFO_STRING` zu übertragen.

für die Funktion [SymbolInfoInteger\(\)](#)

### ENUM\_SYMBOL\_INFO\_INTEGER

| Identifikator                           | Beschreibung                                                                                                                                  | Typ der Eigenschaft                    |
|-----------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------|
| <code>SYMBOL_SELECT</code>              | Symbol ist in Market Watch gewählt                                                                                                            | bool                                   |
| <code>SYMBOL_SESSION_DEALS</code>       | Anzahl der Deals in der aktuellen Session                                                                                                     | long                                   |
| <code>SYMBOL_SESSION_BUY_ORDERS</code>  | Die Gesamtzahl der Buy-Orders im Moment                                                                                                       | long                                   |
| <code>SYMBOL_SESSION_SELL_ORDERS</code> | Die Gesamtzahl der Sell-Orders im Moment                                                                                                      | long                                   |
| <code>SYMBOL_VOLUME</code>              | Volume - Volumen im letzten Deal                                                                                                              | long                                   |
| <code>SYMBOL_VOLUMEHIGH</code>          | Maximales Volumen pro Tag                                                                                                                     | long                                   |
| <code>SYMBOL_VOLUMELOW</code>           | Minimales Volumen pro Tag                                                                                                                     | long                                   |
| <code>SYMBOL_TIME</code>                | Zeit der letzten Quotation                                                                                                                    | datetime                               |
| <code>SYMBOL_DIGITS</code>              | Anzahl der Dezimalzeichen                                                                                                                     | int                                    |
| <code>SYMBOL_SPREAD</code>              | Spreadgröße in Punkten                                                                                                                        | int                                    |
| <code>SYMBOL_SPREAD_FLOAT</code>        | Symbol des fließenden Spreads                                                                                                                 | bool                                   |
| <code>SYMBOL_TICKS_BOOKDEPTH</code>     | Maximale Anzahl der in <a href="#">DOM</a> gezeigten Anforderungen. Für Instrumente, die keine Quelle der Anforderungen haben, ist der Wert 0 | int                                    |
| <code>SYMBOL_TRADE_CALC_MODE</code>     | Mode der Berechnung von Vertragskosten                                                                                                        | <a href="#">ENUM_SYMBOL_CALC_MODE</a>  |
| <code>SYMBOL_TRADE_MODE</code>          | Typ der Orderdurchführung                                                                                                                     | <a href="#">ENUM_SYMBOL_TRADE_MODE</a> |
| <code>SYMBOL_START_TIME</code>          | Anfangsdatum der Versteigerung nach dem Instrument (gewöhnlich wird für Futures verwendet)                                                    | datetime                               |
| <code>SYMBOL_EXPIRATION_TIME</code>     | Abschlussdatum der                                                                                                                            | datetime                               |

|                           |                                                                                          |                                             |
|---------------------------|------------------------------------------------------------------------------------------|---------------------------------------------|
|                           | Versteigerung nach dem Instrument (gewöhnlich wird für Futures verwendet)                |                                             |
| SYMBOL_TRADE_STOPS_LEVEL  | Minimale Einrückung in Punkten vom laufenden Schlusspreis für Einstellung der Stop Order | int                                         |
| SYMBOL_TRADE_FREEZE_LEVEL | Distanz der Einfrierung der Handelsoperationen (in Punkten)                              | int                                         |
| SYMBOL_TRADE_EXEMODE      | Mode des Dealsabschlusses                                                                | <a href="#">ENUM_SYMBOL_TRADE_EXECUTION</a> |
| SYMBOL_SWAP_MODE          | Modell der Swapberechnung                                                                | <a href="#">ENUM_SYMBOL_SWAP_MODE</a>       |
| SYMBOL_SWAP_ROLLOVER3DAYS | Wochentag für Anrechnung des dreifachen Swaps                                            | <a href="#">ENUM_DAY_OF_WEEK</a>            |
| SYMBOL_EXPIRATION_MODE    | Flags der erlaubten <a href="#">Order-Ablaufmodi</a>                                     | int                                         |
| SYMBOL_FILLING_MODE       | Flags der erlaubten <a href="#">Order-Füllmodi</a>                                       | int                                         |
| SYMBOL_ORDER_MODE         | Flags der erlaubten <a href="#">Ordertypen</a>                                           | int                                         |
| SYMBOL_OPTION_MODE        | Тип опциона                                                                              | <a href="#">ENUM_SYMBOL_OPTION_MODE</a>     |
| SYMBOL_OPTION_RIGHT       | Право опциона (Call/Put)                                                                 | <a href="#">ENUM_SYMBOL_OPTION_RIGHT</a>    |

für die Funktion [SymbolInfoDouble\(\)](#)

#### ENUM\_SYMBOL\_INFO\_DOUBLE

| Identifikator        | Beschreibung                    | Typ der Eigenschaft |
|----------------------|---------------------------------|---------------------|
| SYMBOL_BID           | Bid - das beste Verkaufsangebot | double              |
| SYMBOL_BIDHIGH       | maximales Bid pro Tag           | double              |
| SYMBOL_BIDLOW        | Minimales Bid pro Tag           | double              |
| SYMBOL_ASK           | Ask - das beste Kaufsangebot    | double              |
| SYMBOL_ASKHIGH       | Maximales Ask pro Tag           | double              |
| SYMBOL_ASKLOW        | Minimales Ask pro Tag           | double              |
| SYMBOL_LAST          | Preis des letzten Deals         | double              |
| SYMBOL_LASTHIGH      | Maximales Last pro Tag          | double              |
| SYMBOL_LASTLOW       | Minimales Last pro Tag          | double              |
| SYMBOL_OPTION_STRIKE |                                 |                     |



|                                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |        |
|------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------|
| SYMBOL_POINT                       | Wert eines Punktes                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | double |
| SYMBOL_TRADE_TICK_VALUE            | Wert<br>SYMBOL_TRADE_TICK_VALUE_<br>PROFIT                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | double |
| SYMBOL_TRADE_TICK_VALUE_P<br>ROFIT | Berechnete Kosten des Ticks<br>für wirtschaftliche Position                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | double |
| SYMBOL_TRADE_TICK_VALUE_L<br>OSS   | Berechnete Kosten des Ticks<br>für unwirtschaftliche Position                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | double |
| SYMBOL_TRADE_TICK_SIZE             | Minimale Preisveränderung                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | double |
| SYMBOL_TRADE_CONTRACT_SI<br>ZE     | Größe des Handelsvertrags                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | double |
| SYMBOL_VOLUME_MIN                  | Minimale Volumen für<br>Dealsabschluss                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | double |
| SYMBOL_VOLUME_MAX                  | Maximales Volumen für<br>Dealsabschluss                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | double |
| SYMBOL_VOLUME_STEP                 | Minimaler Schritt der<br>Volumenveränderung für<br>Dealsabschluss                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | double |
| SYMBOL_VOLUME_LIMIT                | Die maximale zulässige<br>gesamte Volumen von einer<br>offenen Position und<br>schwebende Ordern in einer<br>Richtung (Kauf oder Verkauf)<br>für das Symbol. Zum Beispiel<br>mit der Begrenzung von 5<br>Lots, können Sie eine offene<br>Kaufposition mit dem Volumen<br>von 5 Lots haben und eine<br>schwebende Order Sell Limit<br>mit dem Volumen von 5 Lots<br>stellen. Aber in diesem Fall<br>können Sie nicht Buy Limit<br>(seit dem Gesamtvolumen in<br>eine Richtung wird die<br>Begrenzung überschreiten)<br>oder Sell Limit mit dem<br>Volumen von mehr als 5 Lots<br>stellen. | double |
| SYMBOL_SWAP_LONG                   | Swapwert beim Kauf                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | double |
| SYMBOL_SWAP_SHORT                  | Swapwert beim Verkauf                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | double |
| SYMBOL_MARGIN_INITIAL              | Initiale Marge bezeichnet die<br>Größe der erforderlichen<br>Kautionssumme in Marge-<br>Währung für Eröffnung einer                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | double |

|                                   |                                                                                                                                                                                                                                                                                                                 |        |
|-----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------|
|                                   | Position im Volumen von einem Lot. Wird verwendet bei der Überprüfung von Geld eines des Kunden beim Eintritt in den Markt.                                                                                                                                                                                     |        |
| SYMBOL_MARGIN_MAINTENANCE         | Maintenance-Marge. Wenn sie eingegeben ist, setzt sie die Marge in der Marge-Währung des Symbols, die von einem Lot abgezogen wird. Wird verwendet bei der Überprüfung von Geld eines des Kunden, wenn sein/ihr Kontozustand ändert. Wenn die Maintenance-Marge gleich 0 ist, wird die Initial Marge verwendet. | double |
| SYMBOL_SESSION_VOLUME             | Das Gesamtvolumen der Deals in der aktuellen Session                                                                                                                                                                                                                                                            | double |
| SYMBOL_SESSION_TURNOVER           | Der Gesamtumsatz in der aktuellen Session                                                                                                                                                                                                                                                                       | double |
| SYMBOL_SESSION_INTEREST           | Der Gesamtumsatz der offenen Positionen                                                                                                                                                                                                                                                                         | double |
| SYMBOL_SESSION_BUY_ORDERS_VOLUME  | Das Gesamtvolumen der Buy-Ordern im Moment                                                                                                                                                                                                                                                                      | double |
| SYMBOL_SESSION_SELL_ORDERS_VOLUME | Das Gesamtvolumen der Sell-Ordern im Moment                                                                                                                                                                                                                                                                     | double |
| SYMBOL_SESSION_OPEN               | Der Eröffnungskurs der Session                                                                                                                                                                                                                                                                                  | double |
| SYMBOL_SESSION_CLOSE              | Der Schlusskurs der Session                                                                                                                                                                                                                                                                                     | double |
| SYMBOL_SESSION_AW                 | Der durchschnittliche Preis für die Session                                                                                                                                                                                                                                                                     | double |
| SYMBOL_SESSION_PRICE_SETTLEMENT   | Abrechnungspreis der aktuellen Session                                                                                                                                                                                                                                                                          | double |
| SYMBOL_SESSION_PRICE_LIMIT_MIN    | Der minimale Preiswert für die Session                                                                                                                                                                                                                                                                          | double |
| SYMBOL_SESSION_PRICE_LIMIT_MAX    | Der maximale Preiswert für die Session                                                                                                                                                                                                                                                                          | double |

für die Funktion [SymbolInfoString\(\)](#)

ENUM\_SYMBOL\_INFO\_STRING

| Identifikator | Beschreibung | Typ der Eigenschaft |
|---------------|--------------|---------------------|
|---------------|--------------|---------------------|

|                        |                                                                                                                                                                                                                                                                                                                                          |        |
|------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------|
| SYMBOL_BASIS           | Имя базового актива для производного инструмента                                                                                                                                                                                                                                                                                         | string |
| SYMBOL_CURRENCY_BASE   | Grundwährung des Instrumentes                                                                                                                                                                                                                                                                                                            | string |
| SYMBOL_CURRENCY_PROFIT | Gewinnwährung                                                                                                                                                                                                                                                                                                                            | string |
| SYMBOL_CURRENCY_MARGIN | Währung der Marge                                                                                                                                                                                                                                                                                                                        | string |
| SYMBOL_BANK            | Quelle der laufenden Quotation                                                                                                                                                                                                                                                                                                           | string |
| SYMBOL_DESCRIPTION     | Zeilenbeschreibung des Symbols                                                                                                                                                                                                                                                                                                           | string |
| SYMBOL_ISIN            | Der Name eines Symbols im System ISIN (International Securities Identification Number). International Securities Identification Number ist ein 12-stelliger alphanumerischer Code zur eindeutigen Identifizierung eines Wertpapiers. Das Vorhandensein dieser Einstellung des Symbols wird auf der Seite eines Handelsservers ermittelt. | string |
| SYMBOL_PATH            | Pfad im Symbolbaum                                                                                                                                                                                                                                                                                                                       | string |

für jedes finanzielles Instrument können mehrere Modi der Gültigkeitsfrist (Ablauffrist) der wartenden Ordern angegeben werden. Für jeden Modus gibt es eine Flagge, Flaggen können miteinander durch die Operation des logischen **ODER** (|) kombiniert werden, ZB, SYMBOL\_EXPIRATION\_GTC | SYMBOL\_EXPIRATION\_SPECIFIED. Für Prüfung der Zulässigkeit des bestimmten Modus für Instrument muss man das Ergebnis der logischen **UND** (&) mit der Flagge des Modus vergleichen.

Wenn für Symbol die Flagge SYMBOL\_EXPIRATION\_SPECIFIED angegeben wird, kann beim Senden der wartenden Order ihre Gültigkeitsfrist explizit angegeben werden.

| Identifikator               | Wert | Beschreibung                                                                     |
|-----------------------------|------|----------------------------------------------------------------------------------|
| SYMBOL_EXPIRATION_GTC       | 1    | Order ist zeitlich uneingeschränkt gültig, bis zu seiner expliziten Annullierung |
| SYMBOL_EXPIRATION_DAY       | 2    | Order ist bis zum Tagesende gültig                                               |
| SYMBOL_EXPIRATION_SPECIFIED | 4    | Ablauffrist wird in der Order angegeben werden                                   |
| SYMBOL_EXPIRATION_SPECIFIED | 8    | Ablaufdatum wird in der Order                                                    |

|       |  |                  |
|-------|--|------------------|
| D_DAY |  | angegeben werden |
|-------|--|------------------|

Beispiel:

```
//+-----+
//| prüft Zulässigkeit des angegebenen Modus der Expiration |
//+-----+
bool IsExpirationTypeAllowed(string symbol,int exp_type)
{
//--- erhalten wir Wert der Eigenschaft, die zulässige Modi der Ablauffrist beschreibt
    int expiration=(int)SymbolInfoInteger(symbol,SYMBOL_EXPIRATION_MODE);
//--- geben wir true zurück, wenn Modus exp_type zulässig ist
    return((expiration & exp_type)==exp_type);
;}
```

Beim Senden der Order kann die Politik der Ausfüllung des im Handelsbefehl gemeldeten Volumens angegeben werden. Zulässige Varianten der Ordererfüllung für Volumen werden für jedes Symbol in der Tabelle angegeben. Für jedes Instrument kann nicht einen, sondernmehrere Modi durch Kombination der Flaggen gestellt werden. Kombination der Flaggen wird durch Operation der logischen ODER (|) ausgedrückt, ZB, SYMBOL\_FILLING\_FOK|SYMBOL\_FILLING\_IOC. für Prüfung der Zulässigkeit bestimmtes Modus für Instrument muss das Ergebnis der logischen UND (&) mit der Flagge des Modus verglichen werden.

| Ausfüllungspolitik | Identifikator      | Wert | Beschreibung                                                                                                                                                                                                                                                                                                                                                      |
|--------------------|--------------------|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Alles/Nichts       | SYMBOL_FILLING_FOK | 1    | Diese Ausfüllungspolitik bedeutet, dass der Order nur in der angegebenen Volumen ausgefüllt werden kann. Wenn die ausreichende Menge eines Finanzinstruments sich auf dem Markt derzeit nicht präsentieren, wird der Order nicht ausgeführt werden. Das erforderliche Volumen kann aus mehreren Vorschlägen, die derzeit auf dem Markt erhältlich sind, bestehen. |
| Alles/Teil         | SYMBOL_FILLING_IOC | 2    | In diesem Fall verpflichtet sich der                                                                                                                                                                                                                                                                                                                              |

|         |                 |  |                                                                                                                                                                                                                                                                                                                                                                |
|---------|-----------------|--|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|         |                 |  | Händler, einen Deal mit dem maximalen Volumen, das auf dem Markt erhältlich ist, im Bereich in der angegebenen Reihenfolge zu machen. Wenn die volle Ausführung unmöglich ist, wird der Order auf das verfügbare Volumen ausgeführt werden und das übrige Volumen wird storniert. Erlaubnis zur Nutzung von IOC Aufträge wird auf dem Trading-Server bestimmt. |
| Abrufen | Kein Identifier |  | Dieser Modus ist für Markt (Buy und Sell), Limit- und Stop-Limit-Aufträge verwendet, und nur in der Modi "Market execution" und "Exchange execution". Im Falle von Teilausfüllung wird ein Markt- oder Limit-Order mit dem Restvolumen nicht entfernt und arbeitet weiter.                                                                                     |

In "Auf Anfrage" und "Sofort" [Modi der Ausführung](#) wird für Markt-Orders immer Ausfüllungspolitik Alles/nichts, und für Limit-Orders "Return" verwendet. In diesem Fall, wenn die Funktionen [OrderSend](#) oder [OrderSendAsync](#) Order senden, ist Eingabe der Ausfüllungstyp für sie nicht erforderlich.

In Ausführungsmodi "Market" und "Exchange" ist die Ausfüllungspolitik "Return" immer für alle Arten von Orders erlaubt. Erlaubnis von anderen Arten wird durch Eigenschaften SYMBOL\_FILLING\_FOK und SYMBOL\_FILLING\_IOC überprüft.

#### Beispiel:

```
//+-----+
//| prüft Zulässigkeit des angegebenen Modus der Ausfüllung |
//+-----+
bool IsFillingTypeAllowed(string symbol,int fill_type)
{
    //--- erhalten wir den Wert der Eigenschaft, die Modus der Ausfüllung beschreibt
```

```

int filling=(int)SymbolInfoInteger(symbol,SYMBOL_FILLING_MODE);
//--- geben wir true zurück, wenn Modus fill_type erlaubt ist
return((filling & fill_type)==fill_type);
; }

```

Wenn die Funktion OrderSend() [Handelsanfrage](#) sendet, müssen Sie die Art der Bestellung für einige Operationen von [ENUM\\_ORDER\\_TYPE](#) eingeben. Nicht alle Arten von Aufträgen können für ein bestimmtes Finanzinstrument erlaubt sein, das Eigenschaft [SYMBOL\\_ORDER\\_MODE](#) beschreibt die Flags der zulässigen Arten von Aufträgen.

| Identifizier            | Wert | Beschreibung                                                        |
|-------------------------|------|---------------------------------------------------------------------|
| SYMBOL_ORDER_MARKET     | 1    | Markt-Orders sind erlaubt (Buy und Sell)                            |
| SYMBOL_ORDER_LIMIT      | 2    | Limit-Orders sind erlaubt (Buy Limit und Sell Limit)                |
| SYMBOL_ORDER_STOP       | 4    | Stop-Orders sind erlaubt (Buy Stop und Sell Stop)                   |
| SYMBOL_ORDER_STOP_LIMIT | 8    | Stop-Limit-Orders sind erlaubt (Buy Stop Limit und Sell Stop Limit) |
| SYMBOL_ORDER_SL         | 16   | Stop Loss ist erlaubt                                               |
| SYMBOL_ORDER_TP         | 32   | Take Profit ist erlaubt                                             |

**Beispiel:**

```
//+-----+
//| Die Funktion druckt Ordertypen, die für den Symbol erlaubt sind |
//+-----+
void Check_SYMBOL_ORDER_MODE(string symbol)
{
//--- das Wert der Eigenschaft, die de erlaubten Ordertypen beschreibt
int symbol_order_mode=(int)SymbolInfoInteger(symbol,SYMBOL_ORDER_MODE);
//--- Prüfung von Markt-Orders (Market Execution)
if((SYMBOL_ORDER_MARKET&symbol_order_mode)==SYMBOL_ORDER_MARKET)
Print(symbol+": Markt-Orders sind erlaubt (Buy und Sell)");
//--- Prüfung von Limit-Orders
if((SYMBOL_ORDER_LIMIT&symbol_order_mode)==SYMBOL_ORDER_LIMIT)
Print(symbol+": Orders Buy Limit und Sell Limit sind erlaubt");
//--- Prüfung von Stop-Orders
if((SYMBOL_ORDER_STOP&symbol_order_mode)==SYMBOL_ORDER_STOP)
Print(symbol+": Orders Buy Stop und Sell Stop sind erlaubt");
//--- Prüfung von Stop Limit Orders
if((SYMBOL_ORDER_STOP_LIMIT&symbol_order_mode)==SYMBOL_ORDER_STOP_LIMIT)
Print(symbol+": Orders Buy Stop Limit und Sell Stop Limit sind erlaubt");
//--- Prüfung ob Stop Loss Orders erlaubt sind
if((SYMBOL_ORDER_SL&symbol_order_mode)==SYMBOL_ORDER_SL)
Print(symbol+": Orders Stop Loss sind erlaubt");
//--- Prüfung ob Take Profit Orders erlaubt sind
if((SYMBOL_ORDER_TP&symbol_order_mode)==SYMBOL_ORDER_TP)
Print(symbol+": Orders Take Profit sind erlaubt");
//---
}
```

für die Erhaltung der Information über Methode der Berechnung der Pfandgeldwerte für das Instrument (Größe der Margeanforderungen) ist die Aufzählung ENUM\_SYMBOL\_CALC\_MODE bestimmt.

#### ENUM\_SYMBOL\_CALC\_MODE

| Identifikator            | Beschreibung                                               | Formel                                                                                                                      |
|--------------------------|------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| SYMBOL_CALC_MODE_FOREX   | Forex mode - Berechnung von Gewinn und Marge für Forex     | Margin: Lots*Contract_Size/<br>Leverage<br>Profit: (close_price-<br>open_price)<br>*Contract_Size*Lots                      |
| SYMBOL_CALC_MODE_FUTURES | Futures mode - Berechnung von Marge und Gewinn für futures | Margin: Lots<br>*InitialMargin*Percentage/10<br>0<br>Profit: (close_price-<br>open_price)*TickPrice/<br>TickSize*Lots       |
| SYMBOL_CALC_MODE_CFD     | CFD mode - Berechnung von Marge und Gewinn für CFD         | Margin: Lots<br>*ContractSize*MarketPrice*Pe<br>rcentage/100<br>Profit: (close_price-<br>open_price)<br>*Contract_Size*Lots |
| SYMBOL_CALC_MODE_CFDIND  | CFD index mode - Berechnung                                | Margin:                                                                                                                     |

|                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                                                                                                                                         |
|-------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| EX                                  | von Marge und Gewinn für CFD durch Indexe                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | $(Lots * ContractSize * MarketPrice) * TickPrice / TickSize$<br>Profit: $(close\_price - open\_price) * Contract\_Size * Lots$          |
| SYMBOL_CALC_MODE_CFDLEVERAGE        | CFD Leverage mode - Berechnung von Marge und Gewinn für CFD beim Leverage-Handel                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | Margin: $(Lots * ContractSize * MarketPrice * Percentage) / Leverage$<br>Profit: $(close\_price - open\_price) * Contract\_Size * Lots$ |
| SYMBOL_CALC_MODE_EXCH_STOCKS        | Exchange mode - Berechnung von Marge und Gewinn für den Handel von Wertpapieren an der Börse                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Margin: $Lots * ContractSize * OpenPrice$<br>Profit: $(close\_price - open\_price) * Contract\_Size * Lots$                             |
| SYMBOL_CALC_MODE_EXCH_FUTURES       | Futures mode - Berechnung von Marge und Gewinn für den Handel von Futures-Kontrakte an der Börse                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | Margin: $Lots * InitialMargin$ oder $Lots * MaintenanceMargin$<br>Profit: $(close\_price - open\_price) * Lots * TickPrice / TickSize$  |
| SYMBOL_CALC_MODE_EXCH_FUTURES_FORTS | FORTS Futures mode - Berechnung von Marge und Gewinn für den Handel von Futures-Kontrakte an FORTS. Marge kann auf die Menge der Abweichung MarginDiscount wie folgt reduziert werden:<br>1. Wenn der Preis für eine lange Position (Kauforder) weniger als der geschätzten Preis ist, $MarginDiscount = Lots * ((PriceSettle - PriceOrder) * TickPrice / TickSize)$<br>2. Wenn der Preis für eine kurze Position (Verkaufsorder) weniger als der geschätzten Preis ist, dann $MarginDiscount = Lots * ((PriceOrder - PriceSettle) * TickPrice / TickSize)$<br>wo:<br>○ PriceSettle - Abrechnungspreis (Clearing) von der vorherigen Session;<br>○ PriceOrder - gewichtete durchschnittliche Preis der Position oder der Eröffnungspreis, der im Order (Anfrage) angegeben ist; | Margin: $Lots * InitialMargin$ oder $Lots * MaintenanceMargin$<br>Profit: $(close\_price - open\_price) * Lots * TickPrice / TickSize$  |



|  |                                                                                                                                                                                                          |  |
|--|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
|  | <ul style="list-style-type: none"> <li>○ TickPrice - Preis von Tick (Kosten von Preisänderungen durch einen Punkt)</li> <li>○ TickSize - die Tick-Größe (minimaler Schritt der Preisänderung)</li> </ul> |  |
|--|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|

Es gibt mehrere Handelsmodi für finanzielle Instrumente. Information über Handelsmodi für bestimmte Instrumente wird in Aufzählungswerten `ENUM_SYMBOL_TRADE_MODE` dargestellt.

#### ENUM\_SYMBOL\_TRADE\_MODE

| Identifikator                            | Beschreibung                                                  |
|------------------------------------------|---------------------------------------------------------------|
| <code>SYMBOL_TRADE_MODE_DISABLED</code>  | Symbolhandel ist verboten                                     |
| <code>SYMBOL_TRADE_MODE_LONGONLY</code>  | Erlaubt werden nur Käufe                                      |
| <code>SYMBOL_TRADE_MODE_SHORTONLY</code> | Erlaubt werden nur Verkäufe                                   |
| <code>SYMBOL_TRADE_MODE_CLOSEONLY</code> | Erlaubt werden die Operationen der Schliessung von Positionen |
| <code>SYMBOL_TRADE_MODE_FULL</code>      | Keine Einschränkungen für Handelsoperationen                  |

Mögliche Modi der Dealsabschlüsse für bestimmte Instrumente sind in der Aufzählung `ENUM_SYMBOL_TRADE_EXECUTION` bestimmt.

#### ENUM\_SYMBOL\_TRADE\_EXECUTION

| Identifikator                                | Beschreibung                    |
|----------------------------------------------|---------------------------------|
| <code>SYMBOL_TRADE_EXECUTION_REQUEST</code>  | Handel auf Anfrage              |
| <code>SYMBOL_TRADE_EXECUTION_INSTANT</code>  | Handel mit laufenden Preisen    |
| <code>SYMBOL_TRADE_EXECUTION_MARKET</code>   | Orderdurchführung auf dem Markt |
| <code>SYMBOL_TRADE_EXECUTION_EXCHANGE</code> | Börslicher Ausführung           |

Methoden der Swapsanrechnung bei Übertragung der Position werden in der Aufzählung `ENUM_SYMBOL_SWAP_MODE` angegeben. Die Methode der Swapsanrechnung bestimmt die Maßeinheiten der Parameter [SYMBOL\\_SWAP\\_LONG](#) und [SYMBOL\\_SWAP\\_SHORT](#). Zum Beispiel, wenn die Swaps in der Einzahlung-Währung von Kunden angerechnet werden, wird in den Parametern das Volumen des angerechneten Swaps gerade in der Einzahlung-Währung des Kunden angegeben.

#### ENUM\_SYMBOL\_SWAP\_MODE

| Identifikator | Beschreibung |
|---------------|--------------|
|---------------|--------------|

|                                   |                                                                                                                                                                                                                                                                                                                           |
|-----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SYMBOL_SWAP_MODE_DISABLED         | Keine Swaps                                                                                                                                                                                                                                                                                                               |
| SYMBOL_SWAP_MODE_POINTS           | Swaps werden in Punkten angerechnet                                                                                                                                                                                                                                                                                       |
| SYMBOL_SWAP_MODE_CURRENCY_SYMBOL  | Swaps werden in Geld in der Basiswährung des Symbols angerechnet                                                                                                                                                                                                                                                          |
| SYMBOL_SWAP_MODE_CURRENCY_MARGIN  | Swaps werden in Geld in der Margin-Währung des Symbols angerechnet.                                                                                                                                                                                                                                                       |
| SYMBOL_SWAP_MODE_CURRENCY_DEPOSIT | Swaps werden in Geld in der Einzahlung-Währung des Kunden angerechnet                                                                                                                                                                                                                                                     |
| SYMBOL_SWAP_MODE_INTEREST_CURRENT | Swaps werden in den jährlichen Zinsen vom Instrumentpreis zum Zeitpunkt der Berechnung des Swaps (Das Bankjahr wird mit 360 Tagen gerechnet) angerechnet                                                                                                                                                                  |
| SYMBOL_SWAP_MODE_INTEREST_OPEN    | Swaps werden in den jährlichen Zinsen vom Preis der Eröffnung der Position nach Symbol (Das Bankjahr wird mit 360 Tagen gerechnet) angerechnet                                                                                                                                                                            |
| SYMBOL_SWAP_MODE_REOPEN_CURRENT   | Swaps werden durch die Wiedereröffnung der Position angerechnet. Am Ende eines Handelstages wird die Position zwangsmäßig geschlossen. Am nächsten Tag wird die Position nach einem Schlusskurs +/- die angegebene Anzahl von Punkten (in den Parametern SYMBOL_SWAP_LONG und SYMBOL_SWAP_SHORT) wieder geöffnet          |
| SYMBOL_SWAP_MODE_REOPEN_BID       | Swaps werden durch die Wiedereröffnung der Position angerechnet. Am Ende eines Handelstages wird die Position zwangsmäßig geschlossen. Am nächsten Tag wird die Position nach dem aktuellen Geldkurs Bid +/- die angegebene Anzahl von Punkten (in den Parametern SYMBOL_SWAP_LONG und SYMBOL_SWAP_SHORT) wieder geöffnet |

für Andeutung des Wochentages sind Werte der Aufzählung ENUM\_DAY\_OF\_WEEK bestimmt.

#### ENUM\_DAY\_OF\_WEEK

| Identifikator | Beschreibung |
|---------------|--------------|
| SUNDAY        | Sonntag      |
| MONDAY        | Montag       |
| TUESDAY       | Dienstag     |
| WEDNESDAY     | Mittwoch     |

|          |            |
|----------|------------|
| THURSDAY | Donnerstag |
| FRIDAY   | Freitag    |
| SATURDAY | Samstag    |

Опцион - это контракт, который дает право, но не обязанность купить или продать базовый актив (товар, акцию, фьючерс и т.д.) по фиксированной цене в течении жизни опциона или в определенный момент времени. Для описания свойств опционов предназначены перечисления, которые описывают тип опциона и право, который он представляет.

#### ENUM\_SYMBOL\_OPTION\_RIGHT

| Идентификатор            | Описание                                                 |
|--------------------------|----------------------------------------------------------|
| SYMBOL_OPTION_RIGHT_CALL | Опцион, дающий право купить актив по фиксированной цене  |
| SYMBOL_OPTION_RIGHT_PUT  | Опцион, дающий право продать актив по фиксированной цене |

#### ENUM\_SYMBOL\_OPTION\_MODE

| Идентификатор               | Описание                                                                                                                                                                            |
|-----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SYMBOL_OPTION_MODE_AMERICAN | Европейский тип опциона - может быть погашен только в указанную дату (дата истечения срока, дата исполнения, дата погашения)                                                        |
| SYMBOL_OPTION_MODE_EUROPEAN | Американский тип опциона - может быть погашен в любой день до истечения срока опциона. Для такого типа задаётся период, в течение которого покупатель может исполнить данный опцион |

## Information über das Konto

für die Erhaltung der Information über das laufende Konto werden die Funktionen [AccountInfoInteger\(\)](#), [AccountInfoDouble\(\)](#) und [AccountInfoString\(\)](#) verwendet. Als Parameter nehmen diese Funktion die Werte aus den entsprechen Aufzählungen `ENUM_ACCOUNT_INFO` an.

für die Funktion [AccountInfoInteger\(\)](#)

### ENUM\_ACCOUNT\_INFO\_INTEGER

| Identifikator          | Beschreibung                                                | Typ der Eigenschaft                       |
|------------------------|-------------------------------------------------------------|-------------------------------------------|
| ACCOUNT_LOGIN          | Nummer des Kontos                                           | long                                      |
| ACCOUNT_TRADE_MODE     | Handelsweise des Kontos                                     | <a href="#">ENUM_ACCOUNT_TRADE_MODE</a>   |
| ACCOUNT_LEVERAGE       | Größe von Leverage                                          | long                                      |
| ACCOUNT_MARGIN_SO_MODE | Mode des minimal zugänglichen Marge                         | <a href="#">ENUM_ACCOUNT_STOPOUT_MODE</a> |
| ACCOUNT_TRADE_ALLOWED  | <a href="#">Erlaubnis zu handeln</a> für das laufende Konto | bool                                      |
| ACCOUNT_TRADE_EXPERT   | <a href="#">Erlaubnis zu handeln</a> für den Experten       | bool                                      |
| ACCOUNT_LIMIT_ORDERS   | Maximale zulässige Anzahl der geltenden Warteordern         | int                                       |

für die Funktion [AccountInfoDouble\(\)](#)

### ENUM\_ACCOUNT\_INFO\_DOUBLE

| Idenfikator         | Beschreibung                                                                            | Typ der Eigenschaft |
|---------------------|-----------------------------------------------------------------------------------------|---------------------|
| ACCOUNT_BALANCE     | Bilanz des Kontos in der Währung des Deposits                                           | double              |
| ACCOUNT_CREDIT      | Grösse des gegebenen Kredits in der Währung des Deposits                                | double              |
| ACCOUNT_PROFIT      | Grösse des laufenden Gewinns auf dem Konto in der Währung des Deposits                  | double              |
| ACCOUNT_EQUITY      | Wert der Eigenmittel auf dem Konto in der Währung des Deposits                          | double              |
| ACCOUNT_MARGIN      | Wert des reservierten Pfandgeldes in der Währung des Deposits                           | double              |
| ACCOUNT_MARGIN_FREE | Grösse der freien Mittel auf dem Konto in der Währung des Deposits, die für die Öffnung | double              |

|                            |                                                                                                                                                                                                                                          |        |
|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------|
|                            | der Position zugänglich sind                                                                                                                                                                                                             |        |
| ACCOUNT_MARGIN_LEVEL       | Konto-Margenniveau in Prozenten                                                                                                                                                                                                          | double |
| ACCOUNT_MARGIN_SO_CALL     | Margin Call Level. Depending on the set ACCOUNT_MARGIN_SO_MODE is expressed in percents or in the deposit currency. Abhängig vom eingestellten ACCOUNT_MARGIN_SO_MODE wird in Prozenten oder in der Währung des Deposits ausgedrückt     | double |
| ACCOUNT_MARGIN_SO_SO       | Margin stop out level. Depending on the set ACCOUNT_MARGIN_SO_MODE is expressed in percents or in the deposit currency. Abhängig vom eingestellten ACCOUNT_MARGIN_SO_MODE wird in Prozenten oder in der Währung des Deposits ausgedrückt | double |
| ACCOUNT_MARGIN_INITIAL     | Размер средств, зарезервированных на счёте, для обеспечения гарантийной суммы по всем отложенным ордерам                                                                                                                                 | double |
| ACCOUNT_MARGIN_MAINTENANCE | Размер средств, зарезервированных на счёте, для обеспечения минимальной суммы по всем открытым позициям                                                                                                                                  | double |
| ACCOUNT_ASSETS             | Текущий размер активов на счёте                                                                                                                                                                                                          | double |
| ACCOUNT_LIABILITIES        | Текущий размер обязательств на счёте                                                                                                                                                                                                     | double |
| ACCOUNT_COMMISSION_BLOCKED | Текущая сумма заблокированных комиссий по счёту                                                                                                                                                                                          | double |

für die Funktion [AccountInfoString\(\)](#)

#### ENUM\_ACCOUNT\_INFO\_STRING

| Identifikator | Beschreibung | Typ der Eigenschaft |
|---------------|--------------|---------------------|
|---------------|--------------|---------------------|

|                  |                                              |        |
|------------------|----------------------------------------------|--------|
| ACCOUNT_NAME     | Name des Kunden                              | string |
| ACCOUNT_SERVER   | Name des Handelsservers                      | string |
| ACCOUNT_CURRENCY | Depositenwaehrung                            | string |
| ACCOUNT_COMPANY  | Name der Gesellschaft, die das Konto bedient | string |

Es gibt verschiedene Typen von Konten, die auf dem Handelsserver geöffnet werden können. Um den Typ des Kontos, auf dem das MQL5-Programm läuft, zu finden, verwenden Sie ENUM\_ACCOUNT\_TRADE\_MODE.

#### ENUM\_ACCOUNT\_TRADE\_MODE

| Identifikator              | Beschreibung        |
|----------------------------|---------------------|
| ACCOUNT_TRADE_MODE_DEMO    | Demo-Handelskonto   |
| ACCOUNT_TRADE_MODE_CONTEST | Konkurshandelskonto |
| ACCOUNT_TRADE_MODE_REAL    | Realkonto           |

Wenn es nicht genügend Eigenkapital, um offene Positionen zu halten, gibt, geschieht eine Situation der erzwungenen Schließung Stop-Out. Die minimale Marge-Ebene, bei dem Stop-Out eintritt, kann als Prozentsatz oder in Form von Geld angegeben werden. Finden Sie heraus, welcher Modus für dieses Konto eingestellt ist, mit der Aufzählung ENUM\_ACCOUNT\_STOPOUT\_MODE.

#### ENUM\_ACCOUNT\_STOPOUT\_MODE

| Identifikator                | Beschreibung                       |
|------------------------------|------------------------------------|
| ACCOUNT_STOPOUT_MODE_PERCENT | Level wird in Prozenten vorgegeben |
| ACCOUNT_STOPOUT_MODE_MONEY   | Level wird in Geld vorgegeben      |

Ein Beispiel für ein Skript, das kurze Information über das Konto anzeigt.

```
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    //-- Firmenname
    string company=AccountInfoString(ACCOUNT_COMPANY);
    //-- Kundename
    string name=AccountInfoString(ACCOUNT_NAME);
    //-- Kontonummer
    long login=AccountInfoInteger(ACCOUNT_LOGIN);
    //-- Servername
    string server=AccountInfoString(ACCOUNT_SERVER);
    //-- Kontowährung
    string currency=AccountInfoString(ACCOUNT_CURRENCY);
    //-- Demo-, Konkurs- oder Realkonto
```

```

ENUM_ACCOUNT_TRADE_MODE account_type=(ENUM_ACCOUNT_TRADE_MODE)AccountInfoInteger(AC
//--- Nun verwandeln wir den Enum-Wert in übersichtlicher Form
string trade_mode;
switch(account_type)
{
    case ACCOUNT_TRADE_MODE_DEMO:
        trade_mode="Demo";
        break;
    case ACCOUNT_TRADE_MODE_CONTEST:
        trade_mode="Konkurs";
        break;
    default:
        trade_mode="Real";
        break;
};
//--- Stop Out wird in Prozent oder in Form von Geld angegeben
ENUM_ACCOUNT_STOPOUT_MODE stop_out_mode=(ENUM_ACCOUNT_STOPOUT_MODE)AccountInfoInteger
//--- Erhalten die Werte der Ebenen, auf denen Margin Call und Stop-Out geschehen
double margin_call=AccountInfoDouble(ACCOUNT_MARGIN_SO_CALL);
double stop_out=AccountInfoDouble(ACCOUNT_MARGIN_SO_SO);
//--- Zeigen kurze Informationen über das Konto
PrintFormat("Konto des Kunden '%s' #d %s öffnet in '%s' am Server '%s'",
            name,login,trade_mode,company,server);
PrintFormat("Kontowährung - %s, MarginCall und StopOut werden in %s" angegeben,
            currency,(stop_out_mode==ACCOUNT_STOPOUT_MODE_PERCENT)?"Prozent":"Geld");
PrintFormat("MarginCall=%G, StopOut=%G",margin_call,stop_out);
};

```

## Teststatistik

Nach dem Test sind verschiedene Parameter der Statistik der Trading-Ergebnisse berechnet. Sie können Werte der Parameter mit Hilfe der Funktion [TesterStatistics\(\)](#) abrufen, mit der Angabe der Parameter-ID aus ENUM\_STATISTICS.

Obwohl zwei Typen von Parametern (int und double) sind für die Berechnung der Statistik verwendet, gibt die Funktion alle Werte in der double-Form zurück. Alle statistischen Werte der double-Typs sind in der Einzahlung-Währung standardmäßig ausgedrückt, sofern nicht anders angegeben.

### ENUM\_STATISTICS

| ID                       | Beschreibung des statistischen Parameters                                                                                    | Typ    |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------|--------|
| STAT_INITIAL_DEPOSIT     | Der Wert des Ersteinzahlung                                                                                                  | double |
| STAT_WITHDRAWAL          | Von einem Konto abgezogene Geld                                                                                              | double |
| STAT_PROFIT              | Nettogewinn nach Test, Betrag von STAT_GROSS_PROFIT und STAT_GROSS_LOSS (STAT_GROSS_LOSS ist immer kleiner oder gleich Null) | double |
| STAT_GROSS_PROFIT        | Total Gewinn, Betrag aller profitablen (positiven) Trades. Der Wert ist größer oder gleich Null                              | double |
| STAT_GROSS_LOSS          | Total Verlust, Betrag aller verlustbringenden (negativen) Trades. Der Wert ist kleiner oder gleich Null                      | double |
| STAT_MAX_PROFITTRADE     | Maximaler Gewinn - der höchste Wert unter allen profitablen Trades. Der Wert ist größer oder gleich Null                     | double |
| STAT_MAX_LOSSTRADE       | Maximaler Verlust - der kleinste Wert unter allen verlustbringenden Trades. Der Wert ist kleiner oder gleich Null            | double |
| STAT_CONPROFITMAX        | Maximaler Gewinn unter konsequent profitable Trades. Der Wert ist größer oder gleich Null                                    | double |
| STAT_CONPROFITMAX_TRADES | Anzahl der Trades, die STAT_CONPROFITMAX geformt haben (maximaler Gewinn unter konsequent profitable                         | int    |



|                            |                                                                                                                                                                                                                  |        |
|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------|
|                            | Trades)                                                                                                                                                                                                          |        |
| STAT_MAX_CONWINS           | Total Gewinn in der längsten Reihe von profitablen Trades                                                                                                                                                        | double |
| STAT_MAX_CONPROFIT_TRADES  | Anzahl der Trades in der längste Reihe von profitablen Trades STAT_MAX_CONWINS                                                                                                                                   | int    |
| STAT_CONLOSSMAX            | Maximaler Verlust unter konsequent profitable Trades. Der Wert ist kleiner oder gleich Null                                                                                                                      | double |
| STAT_CONLOSSMAX_TRADES     | Anzahl der Trades, die STAT_CONLOSSMAX geformt haben (maximaler Verlust unter konsequent verlustbringende Trades)                                                                                                | int    |
| STAT_MAX_CONLOSSES         | Total Verlust in der längsten Reihe von verlustbringenden Trades                                                                                                                                                 | double |
| STAT_MAX_CONLOSS_TRADES    | Anzahl der Trades in der längste Reihe von verlustbringenden Trades STAT_MAX_CONLOSSES                                                                                                                           | int    |
| STAT_BALANCEMIN            | Der minimale Wert der Bilanz                                                                                                                                                                                     | double |
| STAT_BALANCE_DD            | Maximaler Drawdown der Bilanz in Geld. Im Prozeß der Trading kann ein Bilanz zahlreiche Drawdowns haben, hier nehmen wir den größte Wert                                                                         | double |
| STAT_BALANCEDD_PERCENT     | Drawdown der Bilanz als Prozentsatz, der zum Zeitpunkt von maximalem Bilanz-Drawdown in Geld aufgenommen wurde (STAT_BALANCE_DD)                                                                                 | double |
| STAT_BALANCE_DDREL_PERCENT | Maximaler Drawdown der Bilanz als Prozentsatz. Im Prozeß der Trading kann ein Bilanz zahlreiche Drawdowns haben, für jeden Drawdown wird sein Wert als Prozentsatz berechnet. Der größte Wert wird zurückgegeben | double |
| STAT_BALANCE_DD_RELATIVE   | Drawdown der Bilanz in Geld, der zum Zeitpunkt von                                                                                                                                                               | double |

|                           |                                                                                                                                                                                                                               |        |
|---------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------|
|                           | maximalem Bilanz-Drawdown in Prozent aufgenommen wurde<br>(STAT_BALANCE_DDREL_PERCENT)                                                                                                                                        |        |
| STAT_EQUITYMIN            | Der minimale Wert des Eigenkapitals                                                                                                                                                                                           | double |
| STAT_EQUITY_DD            | Maximaler Drawdown des Eigenkapitals in Geld. Im Prozeß der Trading kann ein Eigenkapital zahlreiche Drawdowns haben, hier nehmen wir den größte Wert                                                                         | double |
| STAT_EQUITYDD_PERCENT     | Drawdown des Eigenkapitals in Prozent, der zum Zeitpunkt von maximalem Eigenkapitaldrawdown in Geld aufgenommen wurde<br>(STAT_EQUITY_DD)                                                                                     | double |
| STAT_EQUITY_DDREL_PERCENT | Maximaler Drawdown des Eigenkapitals als Prozentsatz. Im Prozeß der Trading kann ein Eigenkapital zahlreiche Drawdowns haben, für jeden Drawdown wird sein Wert als Prozentsatz berechnet. Der größte Wert wird zurückgegeben | double |
| STAT_EQUITY_DD_RELATIVE   | Drawdown des Eigenkapitals in Geld, der zum Zeitpunkt von maximalem Eigenkapitaldrawdown in Prozent aufgenommen wurde<br>(STAT_EQUITY_DDREL_PERCENT).                                                                         | double |
| STAT_EXPECTED_PAYOFF      | Erwartete Auszahlung                                                                                                                                                                                                          | double |
| STAT_PROFIT_FACTOR        | Profit-Faktor, gleich dem Verhältnis von<br>$\text{STAT\_GROSS\_PROFIT} / \text{STAT\_GROSS\_LOSS}$ . Wenn $\text{STAT\_GROSS\_LOSS}=0$ , ist Profit-Faktor gleich <a href="#">DBL_MAX</a>                                    | double |
| STAT_RECOVERY_FACTOR      | Recovery-Faktor, gleich dem Verhältnis von $\text{STAT\_PROFIT} / \text{STAT\_BALANCE\_DD}$                                                                                                                                   | double |
| STAT_SHARPE_RATIO         | Sharpe Ratio                                                                                                                                                                                                                  | double |

|                          |                                                                                                                      |        |
|--------------------------|----------------------------------------------------------------------------------------------------------------------|--------|
| STAT_MIN_MARGINLEVEL     | Der minimale Wert der Margeebene                                                                                     | double |
| STAT_CUSTOM_ONTESTER     | Der Wert des berechneten Benutzer-Optimierungskriterium, der <a href="#">OnTester()</a> -Funktion zurückgegeben wird | double |
| STAT_DEALS               | Anzahl der Deals                                                                                                     | int    |
| STAT_TRADES              | Anzahl der Trades                                                                                                    | int    |
| STAT_PROFIT_TRADES       | Gewinn-Trades                                                                                                        | int    |
| STAT_LOSS_TRADES         | Verlust-Trades                                                                                                       | int    |
| STAT_SHORT_TRADES        | Kurze Trades                                                                                                         | int    |
| STAT_LONG_TRADES         | Lange Trades                                                                                                         | int    |
| STAT_PROFIT_SHORTTRADES  | Kurze Gewinn-Trades                                                                                                  | int    |
| STAT_PROFIT_LONGTRADES   | Lange Gewinn-Trades                                                                                                  | int    |
| STAT_PROFITTRADES_AVGCON | Die durchschnittliche Länge einer Reihe von Gewinn-Trades                                                            | int    |
| STAT_LOSSTRADES_AVGCON   | Die durchschnittliche Länge einer Reihe von Verlust-Trades                                                           | int    |

## Handelskonstanten

Verschiedenartige Konstanten, die für Programmieren von Handelsstrategien verwendet werden, sind in folgende Gruppen aufgeteilt:

- [Information über historische Daten des Instrumentes](#) - Erhaltung der allgemeinen Information über das finanzielle Instrument;
- [Ordereigenschaften](#) - Erhalten von Information über Handelsordern;
- [Positioneigenschaften](#) - Erhalten von Information über laufende Positionen;
- [Dealeigenschaften](#) - Erhalten von Information über abgeschlossene Deals;
- [Typen der Handelsoperationen](#) - Beschreibung der zugeänglichen Handelsoperationen;
- [Typen der Handelstransaktionen](#) - Beschreibung der möglichen Typen der Handelstransaktionen;
- [Handelsordern in DOM](#) - Trennung der Ordern nach der Richtung der angeforderten Handelsoperation.

## Information über historische Daten des Instrumentes

Beim [Zugang zu Zeitreihen](#) wird für Erhaltung der zusätzlichen [Information über das Instrument](#) die Funktion [SeriesInfoInteger\(\)](#) verwendet. Als Parameter dieser Funktion wird Identifikator der angeforderten Eigenschaft übertragen, der einen der Aufzählungswerte `ENUM_SERIES_INFO_INTEGER` annehmen kann

### ENUM\_SERIES\_INFO\_INTEGER

| Identifikator                          | Beschreibung                                                                               | Typ der Eigenschaft |
|----------------------------------------|--------------------------------------------------------------------------------------------|---------------------|
| <code>SERIES_BARS_COUNT</code>         | Anzahl der Bars für Symbol-Periode zum jetzigen Zeitpunkt                                  | long                |
| <code>SERIES_FIRSTDATE</code>          | Das erste Datum für Symbol-Periode zum jetzigen Zeitpunkt                                  | datetime            |
| <code>SERIES_LASTBAR_DATE</code>       | Öffnungszeit der letzten Bar für Symbol-Periode                                            | datetime            |
| <code>SERIES_SERVER_FIRSTDATE</code>   | Das erste Datum des Symbols auf dem Server unabhängig von der Periode                      | datetime            |
| <code>SERIES_TERMINAL_FIRSTDATE</code> | Das erste Datum in der Geschichte des Symbols im Kundenterminal unabhängig von der Periode | datetime            |
| <code>SERIES_SYNCHRONIZED</code>       | Flagge der Synchronisierung von Symbol/Periode zum jetzigen Zeitpunkt                      | bool                |

## Ordereigenschaften

Befehlsanweisungen, Handelsoperationen durchzuführen, werden als Order formalisiert. Jede Order hat eine Menge der Eigenschaften für Lesen, die Information über die kann durch die Funktionen [OrderGet...\(\)](#) und [HistoryOrderGet...\(\)](#) erhalten werden.

für die Funktionen [OrderGetInteger\(\)](#) und [HistoryOrderGetInteger\(\)](#)

### ENUM\_ORDER\_PROPERTY\_INTEGER

| Identifikator         | Beschreibung                                                                                                                                                                                                                                                                                                                | Typ                                     |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------|
| ORDER_TIME_SETUP      | Set-up Zeit des Orders                                                                                                                                                                                                                                                                                                      | datetime                                |
| ORDER_TYPE            | Ordertyp                                                                                                                                                                                                                                                                                                                    | <a href="#">ENUM_ORDER_TYPE</a>         |
| ORDER_STATE           | Orderstatus                                                                                                                                                                                                                                                                                                                 | <a href="#">ENUM_ORDER_STATE</a>        |
| ORDER_TIME_EXPIRATION | Ablaufzeit einer Order                                                                                                                                                                                                                                                                                                      | datetime                                |
| ORDER_TIME_DONE       | Durchführungszeit oder Annullieren einer Order                                                                                                                                                                                                                                                                              | datetime                                |
| ORDER_TIME_SETUP_MSC  | Zeitpunkt der Erstellung von Order in Millisekunden seit 01.01.1970                                                                                                                                                                                                                                                         | long                                    |
| ORDER_TIME_DONE_MSC   | Zeit der Ausführung/ Abschiebung des Orders in Millisekunden seit 01.01.1970                                                                                                                                                                                                                                                | long                                    |
| ORDER_TYPE_FILLING    | Durchführungstyp nach dem Rest                                                                                                                                                                                                                                                                                              | <a href="#">ENUM_ORDER_TYPE_FILLING</a> |
| ORDER_TYPE_TIME       | Lebenszeit des Orders                                                                                                                                                                                                                                                                                                       | <a href="#">ENUM_ORDER_TYPE_TIME</a>    |
| ORDER_MAGIC           | Identifikator des Experten, der Order gestellt hat (bestimmt dafür, dass jeder Expert seine eigene unikale Nummer stellt)                                                                                                                                                                                                   | long                                    |
| ORDER_POSITION_ID     | <a href="#">Identifikator der Position</a> , den Order nach seine Durchführung erhält. Jede durchgeführte Order erzeugt einen <a href="#">Deal</a> , das eine neue Position eröffnet oder eine schon existierende <a href="#">Position</a> verändert. Durchgeführte Order erhält eben diesen Identifikator dieser Position. | long                                    |

für die Funktionen [OrderGetDouble\(\)](#) und [HistoryOrderGetDouble\(\)](#)

### ENUM\_ORDER\_PROPERTY\_DOUBLE

| Identifikator         | Beschreibung                                                            | Typ    |
|-----------------------|-------------------------------------------------------------------------|--------|
| ORDER_VOLUME_INITIAL  | Initialvolumen bei der Orderstellung                                    | double |
| ORDER_VOLUME_CURRENT  | eine nicht durchgeführte Order                                          | double |
| ORDER_PRICE_OPEN      | Preis angegeben in Order                                                | double |
| ORDER_SL              | Level Stop Loss                                                         | double |
| ORDER_TP              | Level Take Profit                                                       | double |
| ORDER_PRICE_CURRENT   | Laufender Preis des Ordersymbols                                        | double |
| ORDER_PRICE_STOPLIMIT | Preis der Aufstellung der Limit Order bei Auslösung von StopLimit Order | double |

für die Funktionen [OrderGetString\(\)](#) und [HistoryOrderGetString\(\)](#)

#### ENUM\_ORDER\_PROPERTY\_STRING

| Identifikator | Beschreibung     | Typ    |
|---------------|------------------|--------|
| ORDER_SYMBOL  | Symbol der Order | string |
| ORDER_COMMENT | Orderkommentar   | string |

Beim Senden einer Handelsanforderung durch die Funktion [OrderSend\(\)](#) muss für einige Operationen Ordertyp angegeben werden. Ordertyp wird im Feld `type` der speziellen Struktur [MqlTradeRequest](#) angegeben, und kann verschiedene Werte aus der Aufzählung ENUM\_ORDER\_TYPE annehmen.

#### ENUM\_ORDER\_TYPE

| Identifikator              | Beschreibung                                                                                |
|----------------------------|---------------------------------------------------------------------------------------------|
| ORDER_TYPE_BUY             | Marktkauforder                                                                              |
| ORDER_TYPE_SELL            | Martverkauforder                                                                            |
| ORDER_TYPE_BUY_LIMIT       | Schwebende Order Buy Limit                                                                  |
| ORDER_TYPE_SELL_LIMIT      | Schwebende Order Sell Limit                                                                 |
| ORDER_TYPE_BUY_STOP        | Schwebende Order Buy Stop                                                                   |
| ORDER_TYPE_SELL_STOP       | Schwebende Order Sell Stop                                                                  |
| ORDER_TYPE_BUY_STOP_LIMIT  | Beim Erreichen des Orderpreises wird eine Warteorder Buy Limit zum Preis StopLimit gestellt |
| ORDER_TYPE_SELL_STOP_LIMIT | Beim Erreichen des Orderpreises wird eine Warteorder Sell Limit zum Preis StopLimit         |

|  |          |
|--|----------|
|  | gestellt |
|--|----------|

Jede Order hat einen Status, der seinen Stand beschreibt. Für die Erhaltung der Information verwenden Sie die Funktion [OrderGetInteger\(\)](#) oder [HistoryOrderGetInteger\(\)](#) mit dem Modifikator ORDER\_STATE. Zulaessige Werte werden in der Aufzählung ENUM\_ORDER\_STATE aufbewahren.

#### ENUM\_ORDER\_STATE

| Identifikator              | Beschreibung                                                    |
|----------------------------|-----------------------------------------------------------------|
| ORDER_STATE_STARTED        | Order geprüft, aber vom Broker noch nicht angenommen            |
| ORDER_STATE_PLACED         | Order angenommen                                                |
| ORDER_STATE_CANCELED       | Order ist vom Kunden abgelehnt                                  |
| ORDER_STATE_PARTIAL        | Order ist teilweise durchgeführt                                |
| ORDER_STATE_FILLED         | Order ist vollständig durchgeführt                              |
| ORDER_STATE_REJECTED       | Order abgelehnt                                                 |
| ORDER_STATE_EXPIRED        | Order ist nach Ablauffrist abgelehnt                            |
| ORDER_STATE_REQUEST_ADD    | Order ist im Registrierungsstatus (Aussetzung in Handelssystem) |
| ORDER_STATE_REQUEST_MODIFY | Order ist im Änderungsstatus (Verändern von Parametern)         |
| ORDER_STATE_REQUEST_CANCEL | Order ist im Löschungsstatus (Löschung aus dem Handelssystem)   |

Beim Senden einer Handelsanforderung durch die Funktion [OrderSend\(\)](#) kann für Order Ausfüllungspolitik im Feld *type\_filling* in der speziellen Struktur [MqlTradeRequest](#) vorgegeben werden, zulässig sind Werte aus der Aufzählung ENUM\_ORDER\_TYPE\_FILLING. Für Erhaltung des Werte dieser Eigenschaft verwenden Sie die Funktion [OrderGetInteger\(\)](#) oder [HistoryOrderGetInteger\(\)](#) mit dem Modifikator ORDER\_TYPE\_FILLING.

#### ENUM\_ORDER\_TYPE\_FILLING

| Identifikator     | Beschreibung                                                                                                                                                                                                                                                                                                                                                 |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ORDER_FILLING_FOK | Diese Ausfüllungspolitik bedeutet, dass eine Order nur in der angegebenen Menge gefüllt werden kann. Wenn die notwendige Menge eines Finanzinstruments ist auf dem Markt derzeit nicht verfügbar, wird die Order nicht ausgefüllt werden. Das erforderliche Volumen kann mit mehreren Angeboten, die im Moment auf dem Markt verfügbar sind, gefüllt werden. |



|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ORDER_FILLING_IOC    | Dieser Modus bedeutet, dass der Trader mit einem Deal mit dem maximalen auf dem Markt verfügbaren Volumen innerhalb der Order einverstanden ist. Falls das gesamte Volumen einer Order nicht gefüllt werden kann, wird das verfügbare Volumen davon gefüllt werden und das restliche Volumen wird abgebrochen.                                                                                                                                                                                                                                                                                                                                                                   |
| ORDER_FILLING_RETURN | Dieses Modus wird für die Markt- (ORDER_TYPE_BUY und ORDER_TYPE_SELL), Limit- und Stop-Limit-Orders (ORDER_TYPE_BUY_LIMIT, ORDER_TYPE_SELL_LIMIT, ORDER_TYPE_BUY_STOP_LIMIT und ORDER_TYPE_SELL_STOP_LIMIT) und nur in <a href="#">Modi</a> "Markt Ausführung" und "Börse Ausführung" verwendet. Im Falle einer teilweisen Ausführung wird die Markt- oder Limit-Order mit verbleibende Volumen nicht annulliert, sondern weiter verarbeitet. Für die Aktivierung der Ordnern ORDER_TYPE_BUY_STOP_LIMIT und ORDER_TYPE_SELL_STOP_LIMIT wird eine entsprechende Limit-Order ORDER_TYPE_BUY_LIMIT / ORDER_TYPE_SELL_LIMIT mit Ausfüllungstyp ORDER_FILLING_RETURN erstellt werden. |

Gültigkeitsfrist einer Order kann im Feld *type\_time* der speziellen Struktur [MqlTradeRequest](#) angegeben werden beim Senden der Handelsanforderung durch die Funktion [OrderSend\(\)](#). Zulässig sind Werte aus der Aufzählung `ENUM_ORDER_TYPE_TIME`. Für die Erhaltung des Wertes dieser Eigenschaft verwenden Sie die Funktion [OrderGetInteger\(\)](#) oder [HistoryOrderGetInteger\(\)](#) mit dem Modifikator `ORDER_TYPE_TIME`.

#### ENUM\_ORDER\_TYPE\_TIME

| Identifikator            | Beschreibung                                                                                                                                                |
|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ORDER_TIME_GTC           | Warteorder vor der Ablehnung                                                                                                                                |
| ORDER_TIME_DAY           | Order wird nur innerhalb des Handelstages gültig                                                                                                            |
| ORDER_TIME_SPECIFIED     | Order wird bis zum Ablauffrist gültig                                                                                                                       |
| ORDER_TIME_SPECIFIED_DAY | Order wird bis 00:00 Uhr am angegebenen Datum gültig. Wird diese Zeit nicht auf dem Handelszeit fallen, wird der Ablauf in naher den Handelszeit auftreten. |

## Eigenschaften der Positionen

Das Ergebnis der [Handelsoperationen](#) führt zur Öffnung der Position, Veränderung ihres Volumens und/oder Richtung oder ihr Annullieren. Handelsoperationen werden aufgrund der [Ordern](#) durchgeführt, die durch die Funktion [OrderSend\(\)](#) als [Handelsanordnungen](#) gesendet werden. Für jedes finanzielle [Instrument](#) (Symbol) ist nur eine offene Position möglich. Die Position hat Set der Eigenschaften, die für Lesen durch die Funktionen [PositionGet...\(\)](#) zugänglich sind.

für die Funktion [PositionGetInteger\(\)](#)

### ENUM\_POSITION\_PROPERTY\_INTEGER

| Identifikator            | Beschreibung                                                                                                                                                                                                                     | Typ                                |
|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------|
| POSITION_TIME            | Eröffnungszeit der Position                                                                                                                                                                                                      | datetime                           |
| POSITION_TIME_MSC        | Eröffnungszeit der Position in Millisekunden seit 01.01.1970                                                                                                                                                                     | long                               |
| POSITION_TIME_UPDATE     | Positionsänderungszeit in Sekunden seit 01.01.1970                                                                                                                                                                               | long                               |
| POSITION_TIME_UPDATE_MSC | Positionsänderungszeit in Millisekunden seit 01.01.1970                                                                                                                                                                          | long                               |
| POSITION_TYPE            | Typ der Position                                                                                                                                                                                                                 | <a href="#">ENUM_POSITION_TYPE</a> |
| POSITION_MAGIC           | Magic number für die Position (sehen Sie <a href="#">ORDER_MAGIC</a> )                                                                                                                                                           | long                               |
| POSITION_IDENTIFIER      | Identifikator einer Position ist eine unikale Zahl, die jeder neu geöffneten Position zugeordnet wird und innerhalb ihres ganzen Leben nicht verändert wird. Umlauf der Position verändert den Identifikator der Position nicht. | long                               |

für die Funktion [PositionGetDouble\(\)](#)

### ENUM\_POSITION\_PROPERTY\_DOUBLE

| Identifikator       | Beschreibung                              | Typ    |
|---------------------|-------------------------------------------|--------|
| POSITION_VOLUME     | Volumen der Position                      | double |
| POSITION_PRICE_OPEN | Positionspreis                            | double |
| POSITION_SL         | Level Stop Loss für die offene Position   | double |
| POSITION_TP         | Level Take Profit für die offene Position | double |

|                        |                             |        |
|------------------------|-----------------------------|--------|
| POSITION_PRICE_CURRENT | Laufender Preis des Symbols | double |
| POSITION_COMMISSION    | Kommission                  | double |
| POSITION_SWAP          | Gesamtswap                  | double |
| POSITION_PROFIT        | Laufender Gewinn            | double |

für die Funktion [PositionGetString\(\)](#)

#### ENUM\_POSITION\_PROPERTY\_STRING

| Identifikator    | Beschreibung           | Typ    |
|------------------|------------------------|--------|
| POSITION_SYMBOL  | Symbol der Position    | string |
| POSITION_COMMENT | Kommentar zur Position | string |

Richtung der offenen Position (Kauf oder Verkauf) ist durch Wert von der Aufzählung ENUM\_POSITION\_TYPE bestimmt. Für die Erhaltung des Typs der offenen Position muss die Funktion [PositionGetInteger\(\)](#) mit dem Modifikator POSITION\_TYPE verwendet werden.

#### ENUM\_POSITION\_TYPE

| Identifikator      | Beschreibung |
|--------------------|--------------|
| POSITION_TYPE_BUY  | Kauf         |
| POSITION_TYPE_SELL | Verkauf      |

## Eigenschaften der Deals

Geschäft ist die Spiegelung der Tatsache der [Handelsoperation](#) auf Grund der [Order](#), die Handelsanforderung enthält. Jeder Deal wird durch Eigenschaften beschrieben, die ermöglichen, Information über sie zu bekommen. Für Lesen der Werte der Eigenschaften werden die Funktionen der Art [HistoryDealGet...\(\)](#) verwendet, die die Werte aus den entsprechenden Aufzählungen rückgeben.

für die Funktion [HistoryDealGetInteger\(\)](#)

### ENUM\_DEAL\_PROPERTY\_INTEGER

| Identifikator    | Beschreibung                                                                                                                                                                                                                                                                         | Typ                             |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------|
| DEAL_ORDER       | <a href="#">Order</a> , auf deren Grund der Deal abgeschlossen wurde                                                                                                                                                                                                                 | long                            |
| DEAL_TIME        | Zeit des Dealabschlusses                                                                                                                                                                                                                                                             | datetime                        |
| DEAL_TIME_MSC    | Zeitpunkt der Transaktion in Millisekunden seit 01.01.1970                                                                                                                                                                                                                           | long                            |
| DEAL_TYPE        | Typ des Deals                                                                                                                                                                                                                                                                        | <a href="#">ENUM_DEAL_TYPE</a>  |
| DEAL_ENTRY       | Dealsrichtung - Markteingang, Marktausgang oder Kehrwendung                                                                                                                                                                                                                          | <a href="#">ENUM_DEAL_ENTRY</a> |
| DEAL_MAGIC       | Magic number für Deal (sehen Sie <a href="#">ORDER_MAGIC</a> )                                                                                                                                                                                                                       | long                            |
| DEAL_POSITION_ID | <a href="#">Indetifikator der Position</a> , an deren Öffnung, Veränderung oder Schliessung sich der Deal teilnahm. Jede Position hat ihren unikalen Identifikator, der allen Deals zugeordnet wird, die im Instrument innerhalb des ganzen Lebens der Position abgeschlossen wurde. | long                            |

für die Funktion [HistoryDealGetDouble\(\)](#)

### ENUM\_DEAL\_PROPERTY\_DOUBLE

| Identifikator   | Beschreibung                    | Typ    |
|-----------------|---------------------------------|--------|
| DEAL_VOLUME     | Dealvolumen                     | double |
| DEAL_PRICE      | Dealpreis                       | double |
| DEAL_COMMISSION | Dealkommission                  | double |
| DEAL_SWAP       | Gesamtswap beim Schliessen      | double |
| DEAL_PROFIT     | finanzielles Ergebnis des Deals | double |

für die Funktion [HistoryDealGetString\(\)](#)

#### ENUM\_DEAL\_PROPERTY\_STRING

| Identifikator | Beschreibung       | Typ    |
|---------------|--------------------|--------|
| DEAL_SYMBOL   | Dealssymbol        | string |
| DEAL_COMMENT  | Kommentar zum Deal | string |

Jeder Deal wird durch Typ charakterisiert, zulaessige Werte werden in der Aufzählung ENUM\_DEAL\_TYPE gegeben. Für Erhaltung der Information über den Typ des Deals verwenden Sie die Funktion [HistoryDealGetInteger\(\)](#) mit dem Modifikator DEAL\_TYPE.

#### ENUM\_DEAL\_TYPE

| Identifikator                      | Beschreibung                                                                                                                                                                                                                                                                                                                                                        |
|------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DEAL_TYPE_BUY                      | Kauf                                                                                                                                                                                                                                                                                                                                                                |
| DEAL_TYPE_SELL                     | Verkauf                                                                                                                                                                                                                                                                                                                                                             |
| DEAL_TYPE_BALANCE                  | Bilanz                                                                                                                                                                                                                                                                                                                                                              |
| DEAL_TYPE_CREDIT                   | Kredit                                                                                                                                                                                                                                                                                                                                                              |
| DEAL_TYPE_CHARGE                   | zusätzliche Abzüge                                                                                                                                                                                                                                                                                                                                                  |
| DEAL_TYPE_CORRECTION               | Korrektur                                                                                                                                                                                                                                                                                                                                                           |
| DEAL_TYPE_BONUS                    | Aufzählung des Bonus                                                                                                                                                                                                                                                                                                                                                |
| DEAL_TYPE_COMMISSION               | Zusätzliche Kommissionen                                                                                                                                                                                                                                                                                                                                            |
| DEAL_TYPE_COMMISSION_DAILY         | Kommission, die am Ende des Handelstages angerechnet wird                                                                                                                                                                                                                                                                                                           |
| DEAL_TYPE_COMMISSION_MONTHLY       | Kommission, die am Ende des Monats angerechnet wird                                                                                                                                                                                                                                                                                                                 |
| DEAL_TYPE_COMMISSION_AGENT_DAILY   | Agentenkommission, die am Ende des Handelstages angerechnet wird                                                                                                                                                                                                                                                                                                    |
| DEAL_TYPE_COMMISSION_AGENT_MONTHLY | Agentenkommission, die am Ende des Monats angerechnet wird                                                                                                                                                                                                                                                                                                          |
| DEAL_TYPE_INTEREST                 | Anrechnungen von Zinsen auf freie Mittel                                                                                                                                                                                                                                                                                                                            |
| DEAL_TYPE_BUY_CANCELED             | Abgebrochener Kaufdeal. Es kann eine Situation sein, wenn zuvor abgeschlossener Kaufdeal abgebrochen wird. In solchem Fall ändert sich der Typ des früher abgeschlossenen Deals (DEAL_TYPE_BUY) auf DEAL_TYPE_BUY_CANCELED, und seiner Gewinn/Verlust wird auf Null gesetzt. Zuvor erhaltener Gewinn/Verlust wird auf das Konto mittels gesonderten Bilanzoperation |

|                         |                                                                                                                                                                                                                                                                                                                                                                                                  |
|-------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                         | angerechnet/abgebucht                                                                                                                                                                                                                                                                                                                                                                            |
| DEAL_TYPE_SELL_CANCELED | Abgebrochener Verkaufdeal. Es kann eine Situation sein, wenn zuvor abgeschlossener Verkaufdeal abgebrochen wird. In solchem Fall ändert sich der Typ des zuvor abgeschlossenen Deals (DEAL_TYPE_SELL) auf DEAL_TYPE_SELL_CANCELED, und seiner Gewinn/Verlust wird auf Null gesetzt. Zuvor erhaltener Gewinn/Verlust wird auf das Konto mittels gesonderten Bilanzoperation angerechnet/abgebucht |

Deals unterscheiden sich nicht nur durch den Typ, der in der Aufzählung ENUM\_DEAL\_TYPE vorgegeben wird, sondern auch durch die Methode der Positionsveränderung. Das kann einfach Öffnung der Position sein oder Akkumulation der früher geöffneten Position sein (Markteingang), Positionsschliessen durch den gegensätzlichen Deal mit dem entsprechenden Volumen (Marktausgang) oder Kehrwendung der Position falls entgegengerichteten Deal das Volumen der früher geöffneten Position überdeckt.

Alle diesen Situationen werden durch Werte aus der Aufzählung ENUM\_DEAL\_ENTRY beschrieben. Für die Erhaltung dieser Information über den Deal verwenden Sie die Funktion [HistoryDealGetInteger\(\)](#) mit dem Modifikator DEAL\_ENTRY.

#### ENUM\_DEAL\_ENTRY

| Identifikator    | Beschreibung |
|------------------|--------------|
| DEAL_ENTRY_IN    | Markteingang |
| DEAL_ENTRY_OUT   | Marktausgang |
| DEAL_ENTRY_INOUT | Kehrwendung  |

## Typen der Handelsoperationen

Handel wird durch Senden der Befehlsanweisungen, Positionen zu öffnen und auch Befehlsanweisungen, wartende Order einzustellen, modifizieren und entfernen unter Verwendung der Funktion [OrderSend\(\)](#). Jede Handelsorder deutet auf den Typ der angeforderten Handelsoperation. Handelsoperationen werden in der Aufzählung `ENUM_TRADE_REQUEST_ACTIONS` beschrieben.

### `ENUM_TRADE_REQUEST_ACTIONS`

| Identifikator                     | Beschreibung                                                                                              |
|-----------------------------------|-----------------------------------------------------------------------------------------------------------|
| <code>TRADE_ACTION_DEAL</code>    | Handelsorder auf den sofortigen Dealabschluss mit den angegebenen Parametern stellen (Marktorder stellen) |
| <code>TRADE_ACTION_PENDING</code> | Handelsorder für Dealabschluss bei den angegebenen Bedingungen (wartende Order) stellen                   |
| <code>TRADE_ACTION_SLTP</code>    | die Werte Stop Loss und Take Profit der offenen Position verändern                                        |
| <code>TRADE_ACTION_MODIFY</code>  | Parameter der früher eingestellten Order verändern                                                        |
| <code>TRADE_ACTION_REMOVE</code>  | Wartende Handelsserver entfernen                                                                          |

## Typen der Handelstransaktionen

Als Ergebnis der Durchführung von bestimmten Aktionen auf dem Handelskonto, ändert sich der Kontostand. Zu solchen Aktionen gehören:

- Senden der Handelsanfrage aus einer beliebigen MQL5-Anwendung in dem Kundenterminal mittels der Funktionen [OrderSend](#) und [OrderSendAsync](#) und ihre nachfolgende Ausführung;
- Senden der Handelsanfrage über die grafische Oberfläche des Terminals und ihre nachfolgende Ausführung;
- Auslösen von aufgeschobenen Ordnern und Stop-Ordnern auf dem Server;
- Ausführen von Operationen auf der Seite der Handel-Server.

Als Ergebnis dieser Aktionen werden die folgenden Handelstransaktionen durchgeführt:

- Die Verarbeitung der Handelsanfrage;
- Änderung der offenen Aufträge;
- Änderung der Ordergeschichte;
- Änderung der Dealgeschichte;
- Änderung der Position.

Zum Beispiel, beim Senden Marktkauforder, wird Marktkauforder behandelt und eine entsprechende Marktkauforder für das Konto erstellt. Ist die Order ausgeführt, wird sie aus der Auftragsliste entfernt und zur Ordergeschichte hinzugefügt. Dann wird der entsprechende Deal zur Geschichte hinzugefügt und eine neue Position wird erstellt. Alle diese Aktionen sind Handelstransaktionen.

Damit ein Programmierer die Aktionen verfolgen konnte, die bezüglich eines Handelskontos ausgeführt werden, ist die Funktion [OnTradeTransaction](#) vorgesehen. Mit Hilfe dieses Handlers kann man die auf das Konto angewendeten Handelstransaktionen im MQL5-Anwendung bekommen. Beschreibung der Handelstransaktion wird im ersten Parameter [OnTradeTransaction](#) mit Hilfe der Struktur [MqlTradeTransaction](#) übergeben.

Typ der Handelstransaktion wird im type Parameter der Struktur [MqlTradeTransaction](#) übergeben. Möglichen Typen der Handelstransaktionen werden von der folgenden Aufzählung beschrieben:

### ENUM\_TRADE\_TRANSACTION\_TYPE

| Identifikator                  | Beschreibung                                                                                                                                                                                                                                                                                                                                                                                                               |
|--------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TRADE_TRANSACTION_ORDER_ADD    | Hinzufügen von einer neuen offenen Order.                                                                                                                                                                                                                                                                                                                                                                                  |
| TRADE_TRANSACTION_ORDER_UPDATE | Änderung der offenen Order. Zu solchen Änderungen gehören nicht nur die offensichtlichen Änderungen seitens des Kundenterminals oder Handel-servers, sondern auch die Änderung ihres Zustandes bei der Ausstellung (zum Beispiel, Übergang vom Zustand <a href="#">ORDER_STATE_STARTED</a> im <a href="#">ORDER_STATE_PLACED</a> oder vom <a href="#">ORDER_STATE_PLACED</a> im <a href="#">ORDER_STATE_PARTIAL</a> usw.). |
| TRADE_TRANSACTION_ORDER_DELETE | Entfernung der Order aus der offenen                                                                                                                                                                                                                                                                                                                                                                                       |



|                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                  | Auftragsliste. Eine Order kann aus der offenen Auftragsliste als Ergebnis der Stellung der entsprechenden Anfrage oder als Ergebnis der Durchführung (Füllung) und Übertragung in die Geschichte entfernt werden.                                                                                                                                                                                                                                                                                                                                                                                                |
| TRADE_TRANSACTION_DEAL_ADD       | Hinzufügen des Deals zur Geschichte. Es erfolgt als Ergebnis der Orderausfüllung oder Durchführung der Operationen mit einem Kontostand.                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| TRADE_TRANSACTION_DEAL_UPDATE    | Änderung des Deals in der Geschichte. Es kann eine Situation sein, wenn zuvor ausgeführter Deal auf dem Server geändert wird. Zum Beispiel, ein Deal war im externen Handelssystem (die Börse) geändert, wohin er vom Broker übertragen wurde.                                                                                                                                                                                                                                                                                                                                                                   |
| TRADE_TRANSACTION_DEAL_DELETE    | Entfernung der Order aus der Geschichte. Es kann eine Situation sein, wenn zuvor ausgeführter Deal vom Server entfernt wird. Zum Beispiel, ein Deal war vom externen Handelssystem (die Börse) entfernt, wohin er vom Broker übertragen wurde.                                                                                                                                                                                                                                                                                                                                                                   |
| TRADE_TRANSACTION_HISTORY_ADD    | Hinzufügen der Order zur Geschichte als Ergebnis der Ausführung oder Stornierung.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| TRADE_TRANSACTION_HISTORY_UPDATE | Änderung der Order, die sich in der Ordergeschichte befindet. Dieser Typ ist für die Erweiterung der Funktionalität auf der Seite des Handel-Servers vorgesehen.                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| TRADE_TRANSACTION_HISTORY_DELETE | Entfernung der Order aus der Ordergeschichte. Dieser Typ ist für die Erweiterung der Funktionalität auf der Seite des Handel-Servers vorgesehen.                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| TRADE_TRANSACTION_POSITION       | Änderung der Position, die mit der Ausführung des Deals nicht verbunden ist. Dieser Typ der Transaktion zeigt, dass die Position auf der Seite des Handel-Servers geändert wurde. Volumen, Eröffnungspreis sowie Stop Loss und Take Profit Ebenen von Position können geändert werden. Informationen über Änderungen werden in der Struktur <a href="#">MqlTradeTransaction</a> durch den Handler OnTradeTransaction übergeben. Änderung der Position (Hinzufügen, Änderung oder Liquidation) als Ergebnis des Abschlusses des Deals zieht kein Erscheinen der Transaktion TRADE_TRANSACTION_POSITION nach sich. |
| TRADE_TRANSACTION_REQUEST        | Benachrichtigung, dass die Handelsanfrage vom                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

|  |                                                                                                                                                                                                                                                                                                                                                                                   |
|--|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  | Server bearbeitet ist und das Ergebnis seiner Bearbeitung empfangen ist. Für Transaktionen dieses Typs in der Struktur <a href="#">MqlTradeTransaction</a> muss nur type Feld (Typ der Transaktion) analysiert werden. Für zusätzliche Informationen muss man die zweiten und dritten Parameter der Funktion <a href="#">OnTradeTransaction</a> (request und result) analysieren. |
|--|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Verschiedene Parameter werden je nach Typ der Handelstransaktion in der Struktur MqlTradeTransaction, die sie beschreibt, ausgefüllt. Die ausführliche Beschreibung der übertragenen Daten ist im Abschnitt "[Struktur der Handelstransaktion](#)" gegeben.

Sehen Sie auch

[Struktur der Handelstransaktion](#), [OnTradeTransaction](#)

## Handelsordern in DOM

für Boersensymbole ist das Fenster "Auftragswarteschlange", zugaenglich, wo laufende Kaufs- und Verkaufsordern zu sehen sind. Für jede Order ist die erwuenschte Richtung der Handelsoperation, das erforderliche Volumen und der angeforderte Preis zu sehen.

für Erhaltung der Information über laufendes DOM durch Mittel der Sprache MQL5 ist die Funktion [MarketBookGet\(\)](#) bestimmt, die "screen shot" ins Feld der Strukturen [MqlBookInfo](#) stellt. Jedes Element dieses Feldes hat im Feld *type* die Information über die Richtung der Order - der Wert aus der Aufzählung `ENUM_BOOK_TYPE`.

### ENUM\_BOOK\_TYPE

| Identifikator         | Beschreibung                 |
|-----------------------|------------------------------|
| BOOK_TYPE_SELL        | Verkaufsorder                |
| BOOK_TYPE_BUY         | Kaufsorder                   |
| BOOK_TYPE_SELL_MARKET | Verkaufsorder mit Marktpreis |
| BOOK_TYPE_BUY_MARKET  | Kaufsorder mit Marktpreis    |

Sehen Sie auch

[Strukturen und Klassen](#), [DOM Struktur](#), [Typen der Handelsoperationen](#), [Marktinformation erhalten](#)

## Свойства сигналов

Значения перечислений для работы с торговыми сигналами и настройками их копирования.

Перечисления свойств типа [double](#) торговых сигналов:

### ENUM\_SIGNAL\_BASE\_DOUBLE

| Константа                | Описание                                        |
|--------------------------|-------------------------------------------------|
| SIGNAL_BASE_BALANCE      | Баланс счета                                    |
| SIGNAL_BASE_EQUITY       | Средства на счете                               |
| SIGNAL_BASE_GAIN         | Прирост счета в процентах                       |
| SIGNAL_BASE_MAX_DRAWDOWN | Максимальная просадка                           |
| SIGNAL_BASE_PRICE        | Цена подписки на сигнал                         |
| SIGNAL_BASE_ROI          | Значение ROI (Return on Investment) сигнала в % |

Перечисления свойств типа [integer](#) торговых сигналов:

### ENUM\_SIGNAL\_BASE\_INTEGER

| Константа                  | Описание                                                    |
|----------------------------|-------------------------------------------------------------|
| SIGNAL_BASE_DATE_PUBLISHED | Дата публикации сигнала (когда стал доступен для подписки)  |
| SIGNAL_BASE_DATE_STARTED   | Дата начала мониторинга сигнала                             |
| SIGNAL_BASE_ID             | ID сигнала                                                  |
| SIGNAL_BASE_LEVERAGE       | Плечо торгового счета                                       |
| SIGNAL_BASE_PIPS           | Результат торговли в пипсах                                 |
| SIGNAL_BASE_RATING         | Позиция в рейтинге сигналов                                 |
| SIGNAL_BASE_SUBSCRIBERS    | Количество подписчиков                                      |
| SIGNAL_BASE_TRADES         | Количество трейдов                                          |
| SIGNAL_BASE_TRADE_MODE     | Тип счета (0-реальный счет, 1-демо-счет, 2-конкурсный счет) |

Перечисления свойств типа [string](#) торговых сигналов:

### ENUM\_SIGNAL\_BASE\_STRING

| Константа                | Описание                        |
|--------------------------|---------------------------------|
| SIGNAL_BASE_AUTHOR_LOGIN | Логин автора сигнала            |
| SIGNAL_BASE_BROKER       | Наименование брокера (компания) |

|                           |                      |
|---------------------------|----------------------|
| SIGNAL_BASE_BROKER_SERVER | Сервер брокера       |
| SIGNAL_BASE_NAME          | Имя сигнала          |
| SIGNAL_BASE_CURRENCY      | Валюта счета сигнала |

Перечисления свойств типа [double](#) настроек копирования торговых сигналов:

#### ENUM\_SIGNAL\_INFO\_DOUBLE

| Константа                  | Описание                                                                                                        |
|----------------------------|-----------------------------------------------------------------------------------------------------------------|
| SIGNAL_INFO_EQUITY_LIMIT   | Процент для конвертации объема сделки                                                                           |
| SIGNAL_INFO_SLIPPAGE       | Величина проскальзывания, с которым выставляются рыночные ордера при синхронизации позиций и копировании сделок |
| SIGNAL_INFO_VOLUME_PERCENT | Значение ограничения по средствам для сигнала, r/o                                                              |

Перечисления свойств типа [integer](#) настроек копирования торговых сигналов:

#### ENUM\_SIGNAL\_INFO\_INTEGER

| Константа                          | Описание                                                       |
|------------------------------------|----------------------------------------------------------------|
| SIGNAL_INFO_CONFIRMATIONS_DISABLED | Флаг разрешения синхронизации без показа диалога подтверждения |
| SIGNAL_INFO_COPY_SLTP              | Флаг копирования Stop Loss и Take Profit                       |
| SIGNAL_INFO_DEPOSIT_PERCENT        | Ограничения по депозиту (в %)                                  |
| SIGNAL_INFO_ID                     | id сигнала, r/o                                                |
| SIGNAL_INFO_SUBSCRIPTION_ENABLED   | Флаг разрешения на копирование сделок по подписке              |
| SIGNAL_INFO_TERMS_AGREE            | Флаг согласия с условиями использования сервиса "Сигналы", r/o |

Перечисления свойств типа [string](#) настроек копирования торговых сигналов:

#### ENUM\_SIGNAL\_INFO\_STRING

| Константа        | Описание         |
|------------------|------------------|
| SIGNAL_INFO_NAME | Имя сигнала, r/o |

Смотри также

[Управление сигналами](#)

## Benannte Konstanten

Alle in der Sprache MQL5 verwendeten Konstanten können in folgende Gruppen aufgeteilt werden:

- [Vorbestimmte Makrosubstitutionen](#) - Werte werden in der Compilierungsetappe substituiert;
- [Mathematische Konstanten](#) - Werte einiger mathematischer Ausdrücke;
- [Konstanten der numerischen Typen](#) - Grenzbedingungen, die in einfache Typen gesteckt werden;
- [Deinitialisierungsgruende](#) - Beschreibung der Deinitialisierungsgruende;
- [Pruefung des Objektanzeigers](#) - Aufzählung der Anzeigertypen, die durch die Funktion [CheckPointer\(\)](#) rückgegeben werden ;
- [Andere Konstanten](#) - alles, was ausser anderer Konstantengruppen bleibt.

## Vorbestimmte Makrosubstitutionen

für die Erleichterung des ebugging und der Informationserhaltung über die Arbeit des mql5-Programms werden spezielle Konstanten-Makros eingeführt, deren Werte im Augenblick der Compilierung vorgegeben werden. Das einfachste Verfahren von Verwendung dieser Konstanten - Ausgabe der Werte mit der Funktion [Print\(\)](#), wie es im Beispiel gezeigt wird.

| Konstante                   | Beschreibung                                                                                                                                                                                                       |
|-----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| __DATE__                    | Datum der Kompilierung einer Date ohne die Uhrzeit (Stunden, Minuten und Sekunden sind 0)                                                                                                                          |
| __DATETIME__                | Datum und Uhrzeit der Kompilierung einer Date                                                                                                                                                                      |
| __LINE__                    | Nummer der Zeile im Ausgangskode, auf der sich der Makro befindet                                                                                                                                                  |
| __FILE__                    | Name der laufenden kompilierten Datei                                                                                                                                                                              |
| __PATH__                    | Absoluter Pfad zu der laufenden kompilierten Datei                                                                                                                                                                 |
| __FUNCTION__                | Funktionsname, in deren Körper sich das Makro befindet                                                                                                                                                             |
| __FUNCSIG__                 | Signatur der Funktion, in deren Körper befindet sich das Makro. Protokollierung der vollständige Beschreibung der Funktionen kann bei der Identifizierung von <a href="#">überladenen Funktionen</a> nützlich sein |
| __MQLBUILD__, __MQL5BUILD__ | Buildnummer des Compilers                                                                                                                                                                                          |

### Beispiel:

```
#property copyright "Copyright © 2009, MetaQuotes Software Corp."
#property link      "http://www.metaquotes.net"

//+-----+
//| Expert initialization function |
//+-----+
void OnInit()
{
    //--- Beispiel der Informationsausgabe bei der Ratgeberinitialisierung
    Print(" __FUNCTION__ = ", __FUNCTION__, " __LINE__ = ", __LINE__ );
    //--- Einstellung des Intervals zwischen Ereignisse des Timers
    EventSetTimer(5);
    //---
};
//+-----+
//| Expert deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
    //--- Beispiel der Inforamtionsausgabe bei der Deinitialisierung des Ratgebers
```

```

    Print(" __FUNCTION__ = ", __FUNCTION__, " __LINE__ = ", __LINE__ );
//---
; }
//+-----+
//| Expert tick function |
//+-----+
void OnTick()
{
//--- Ausgabe der Information beim Tickaufnahme
    Print(" __MQLBUILD__ = ", __MQLBUILD__, " __FILE__ = ", __FILE__ );
    Print(" __FUNCTION__ = ", __FUNCTION__, " __LINE__ = ", __LINE__ );
    test1( __FUNCTION__ );
    test2();
//---
; }
//+-----+
//| test1 |
//+-----+
void test1(string par)
{
//--- Informationsausgabe innerhalb der Funktion
    Print(" __FUNCTION__ = ", __FUNCTION__, " __LINE__ = ", __LINE__, " par = ", par);
; }
//+-----+
//| test2 |
//+-----+
void test2()
{
//--- Informationsausgabe innerhalb der Funktion
    Print(" __FUNCTION__ = ", __FUNCTION__, " __LINE__ = ", __LINE__ );
; }
//+-----+
//| OnTimer event handler |
//+-----+
void OnTimer()
{
//---
    Print(" __FUNCTION__ = ", __FUNCTION__, " __LINE__ = ", __LINE__ );
    test1( __FUNCTION__ );
; }

```



## Mathematische Konstanten

für einige mathematische Funktionen werden spezielle Konstanten reserviert, die Werte enthalten. Diese Konstanten können in jeder Stelle des mql5-Programms statt der Berechnung ihrer Werte durch [mathematische Funktionen](#).

| Konstante  | Beschreibung   | Wert                    |
|------------|----------------|-------------------------|
| M_E        | e              | 2.71828182845904523536  |
| M_LOG2E    | $\log_2(e)$    | 1.44269504088896340736  |
| M_LOG10E   | $\log_{10}(e)$ | 0.434294481903251827651 |
| M_LN2      | $\ln(2)$       | 0.693147180559945309417 |
| M_LN10     | $\ln(10)$      | 2.30258509299404568402  |
| M_PI       | pi             | 3.14159265358979323846  |
| M_PI_2     | $\pi/2$        | 1.57079632679489661923  |
| M_PI_4     | $\pi/4$        | 0.785398163397448309616 |
| M_1_PI     | $1/\pi$        | 0.318309886183790671538 |
| M_2_PI     | $2/\pi$        | 0.636619772367581343076 |
| M_2_SQRTPI | $2/\sqrt{\pi}$ | 1.12837916709551257390  |
| M_SQRT2    | $\sqrt{2}$     | 1.41421356237309504880  |
| M_SQRT1_2  | $1/\sqrt{2}$   | 0.707106781186547524401 |

Beispiel:

```
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    //--- geben wir Werte der Konstanten aus
    Print("M_E = ", DoubleToString(M_E, 16));
    Print("M_LOG2E = ", DoubleToString(M_LOG2E, 16));
    Print("M_LOG10E = ", DoubleToString(M_LOG10E, 16));
    Print("M_LN2 = ", DoubleToString(M_LN2, 16));
    Print("M_LN10 = ", DoubleToString(M_LN10, 16));
    Print("M_PI = ", DoubleToString(M_PI, 16));
    Print("M_PI_2 = ", DoubleToString(M_PI_2, 16));
    Print("M_PI_4 = ", DoubleToString(M_PI_4, 16));
    Print("M_1_PI = ", DoubleToString(M_1_PI, 16));
    Print("M_2_PI = ", DoubleToString(M_2_PI, 16));
    Print("M_2_SQRTPI = ", DoubleToString(M_2_SQRTPI, 16));
    Print("M_SQRT2 = ", DoubleToString(M_SQRT2, 16));
    Print("M_SQRT1_2 = ", DoubleToString(M_SQRT1_2, 16));
}
```

```
; }
```

## Konstanten der numerischen Datentypen

Jeder einfache numerische Typ ist für gewisse Aufgaben bestimmt und ermöglicht Arbeit des mql5-Programms beim richtigen Verwenden zu optimieren. Für bessere Kodelesbarkeit und richtige Bearbeitung der Berechnungsergebnisse sind Konstanten eingeführt, die Informationen über Einschränkungen zu bekommen ermöglichen, durch die einer oder anderer Typ einfacher Daten bechraenkt werden.

| Konstante  | Beschreibung                                                     | Wert                 |
|------------|------------------------------------------------------------------|----------------------|
| CHAR_MIN   | Minimaler Wert, der durch den Typ char dargestellt werden kann   | -128                 |
| CHAR_MAX   | Maximaler Wert, der durch den Typ char dargestellt werden kann   | 127                  |
| UCHAR_MAX  | Maximaler Wert, der durch den Typ uchar dargestellt werden kann  | 255                  |
| SHORT_MIN  | Minimaler Wert, der durch den Typ short dargestellt werden kann  | -32768               |
| SHORT_MAX  | Maximaler Wert, der durch den Typ short dargestellt werden kann  | 32767                |
| USHORT_MAX | Maximaler Wert, der durch den Typ ushort dargestellt werden kann | 65535                |
| INT_MIN    | Minimaler Wert, der durch den Typ int dargestellt werden kann    | -2147483648          |
| INT_MAX    | Maximaler Wert, der durch den Typ int dargestellt werden kann    | 2147483647           |
| UINT_MAX   | Maximaler Wert, der durch den Typ uint dargestellt werden kann   | 4294967295           |
| LONG_MIN   | Minimaler Wert, der durch den Typ long dargestellt werden kann   | -9223372036854775808 |
| LONG_MAX   | Maximaler Wert, der durch den Typ long dargestellt werden kann   | 9223372036854775807  |
| ULONG_MAX  | Maximaler Wert, der durch den Typ ulong dargestellt              | 18446744073709551615 |

|                |                                                                                            |                         |
|----------------|--------------------------------------------------------------------------------------------|-------------------------|
|                | werden kann                                                                                |                         |
| DBL_MIN        | Minimaler positiver Wert, der durch den Typ double dargestellt werden kann                 | 2.2250738585072014e-308 |
| DBL_MAX        | Maximaler Wert, der durch den Typ double dargestellt werden kann                           | 1.7976931348623158e+308 |
| DBL_EPSILON    | Der minimale Wert, für den Die Bedingung $1.0 + \text{DBL\_EPSILON} \neq 1.0$ erfüllt wird | 2.2204460492503131e-016 |
| DBL_DIG        | Anzahl der bedeutenden Dezimalzeichen                                                      | 15                      |
| DBL_MANT_DIG   | Anzahl der Bits in mantissa                                                                | 53                      |
| DBL_MAX_10_EXP | Maximaler dezimaler Wert der Exponentenstufung                                             | 308                     |
| DBL_MAX_EXP    | Maximaler binaerer Wert der Exponentenstufung                                              | 1024                    |
| DBL_MIN_10_EXP | Minimaler dezimaler Wert der Exponentenstufung                                             | (-307)                  |
| DBL_MIN_EXP    | Minimaler binaerer Wert der Exponentenstufung                                              | (-1021)                 |
| FLT_MIN        | Minimaler positiver Wert, der durch den Typ float repräsentiert werden kann                | 1.175494351e-38         |
| FLT_MAX        | Maximaler Wert, der durch den Typ float repräsentiert werden kann                          | 3.402823466e+38         |
| FLT_EPSILON    | Der minimale Wert, für den die Bedingung $1.0 + \text{FLT\_EPSILON} \neq 1.0$ erfüllt wird | 1.192092896e-07         |
| FLT_DIG        | Anzahl der bedeutenden Dezimalzeichen                                                      | 6                       |
| FLT_MANT_DIG   | Anzahl der Bits in mantissa                                                                | 24                      |
| FLT_MAX_10_EXP | Maximaler dezimaler Wert der Exponentenstufung                                             | 38                      |
| FLT_MAX_EXP    | Maximaler binaerer Wert der Exponentenstufung                                              | 128                     |
| FLT_MIN_10_EXP | Minimaler dezimaler Wert der Exponentenstufung                                             | -37                     |

|             |                                              |        |
|-------------|----------------------------------------------|--------|
| FLT_MIN_EXP | Minimaler binärer Wert der Exponentenstufung | (-125) |
|-------------|----------------------------------------------|--------|

**Beispiel:**

```

void OnStart()
{
//--- geben wir Werte der Konstanten aus
printf("CHAR_MIN = %d",CHAR_MIN);
printf("CHAR_MAX = %d",CHAR_MAX);
printf("UCHAR_MAX = %d",UCHAR_MAX);
printf("SHORT_MIN = %d",SHORT_MIN);
printf("SHORT_MAX = %d",SHORT_MAX);
printf("USHORT_MAX = %d",USHORT_MAX);
printf("INT_MIN = %d",INT_MIN);
printf("INT_MAX = %d",INT_MAX);
printf("UINT_MAX = %u",UINT_MAX);
printf("LONG_MIN = %I64d",LONG_MIN);
printf("LONG_MAX = %I64d",LONG_MAX);
printf("ULONG_MAX = %I64u",ULONG_MAX);
printf("EMPTY_VALUE = %.16e",EMPTY_VALUE);
printf("DBL_MIN = %.16e",DBL_MIN);
printf("DBL_MAX = %.16e",DBL_MAX);
printf("DBL_EPSILON = %.16e",DBL_EPSILON);
printf("DBL_DIG = %d",DBL_DIG);
printf("DBL_MANT_DIG = %d",DBL_MANT_DIG);
printf("DBL_MAX_10_EXP = %d",DBL_MAX_10_EXP);
printf("DBL_MAX_EXP = %d",DBL_MAX_EXP);
printf("DBL_MIN_10_EXP = %d",DBL_MIN_10_EXP);
printf("DBL_MIN_EXP = %d",DBL_MIN_EXP);
printf("FLT_MIN = %.8e",FLT_MIN);
printf("FLT_MAX = %.8e",FLT_MAX);
printf("FLT_EPSILON = %.8e",FLT_EPSILON);
; }

```

## Deinitialisierungsgruende

Koden der Deinitialisierungsgruende von [Expert](#), die durch die Funktion [UninitializeReason\(\)](#) zurückgegeben werden, können jede der folgenden Werte haben:

| Konstante          | Wert | Beschreibung                                                                                                                               |
|--------------------|------|--------------------------------------------------------------------------------------------------------------------------------------------|
| REASON_PROGRAM     | 0    | Expert beendet seine Operation durch Aufruf der Funktion <a href="#">ExpertRemove()</a>                                                    |
| REASON_REMOVE      | 1    | Das Programm ist entfernt vom Chart                                                                                                        |
| REASON_RECOMPILE   | 2    | Das Programm ist umcompiliert                                                                                                              |
| REASON_CHARTCHANGE | 3    | Symbol oder Chartperiode wurde verändert                                                                                                   |
| REASON_CHARTCLOSE  | 4    | Das Chart wurde abgeschlossen                                                                                                              |
| REASON_PARAMETERS  | 5    | Eingabeparameter wurden vom Benutzer verändert                                                                                             |
| REASON_ACCOUNT     | 6    | Ein anderes Konto wurde aktiviert oder das Konto wurde mit dem Handelsserver neu verbunden, nachdem die Kontoeinstellungen geändert worden |
| REASON_TEMPLATE    | 7    | andere Chartschablone wurde angewendet                                                                                                     |
| REASON_INITFAILED  | 8    | Der Wert bedeutet, dass Bearbeiter <a href="#">OnInit()</a> Nichtnullwert zurückgegeben hat                                                |
| REASON_CLOSE       | 9    | Das Terminal wurde geschlossen                                                                                                             |

Kode des Deinitialisierungsgrundes wird auch als Parameter der vorbestimmten Funktion übertragen [OnDeinit](#)(const int reason).

Indikatoren erhalten den Kode 1(REASON\_REMOVE)und den Kode 2(REASON\_RECOMPILE).

**Beispiel:**

```
//+-----+
//| get text description |
//+-----+
string getUninitReasonText(int reasonCode)
```

```

{
    string text="";
//---
    switch(reasonCode)
    {
        case REASON_ACCOUNT:
            text="Account was changed";break;
        case REASON_CHARTCHANGE:
            text="Symbol or timeframe was changed";break;
        case REASON_CHARTCLOSE:
            text="Chart was closed";break;
        case REASON_PARAMETERS:
            text="Input-parameter was changed";break;
        case REASON_RECOMPILE:
            text="Program "+__FILE__+" was recompiled";break;
        case REASON_REMOVE:
            text="Program "+__FILE__+" was removed from chart";break;
        case REASON_TEMPLATE:
            text="New template was applied to chart";break;
        default:text="Another reason";
    }
//---
    return text;
;}

//+-----+
//| Expert deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
//--- Zweiter Weg , Kode des Deinitialisierungsgrundes zu bekommen
    Print(__FUNCTION__,"_Kode des Deinitialisierungsgrundes = ",reason);
//--- Zweiter Weg, Kode des Deinitialisierungsgrundes zu bekommen
    Print(__FUNCTION__,"_UninitReason = ",getUninitReasonText(_UninitReason));
;}

```

## Pruefung des Objektanzeigers

für die Pruefung des Typs von [Objektanzeiger](#) ist die Funktion [CheckPointer\(\)](#) vorausbestimmt, die den Wert aus der Aufzählung `ENUM_POINTER_TYPE` rückgibt. Wenn ein unkorrekter Anzeiger verwendet wird, wird die Programmausführung sofort unterbrochen.

Objekte, erzeugt von der Anweisung [new](#), sind Objekte des Typs `POINTER_DYNAMIC`. Ausschliesslich für solche Anzeiger kann und muss man [delete\(\)](#) [Anweisung](#) verwenden.

Alle anderen Anzeiger gehören zum Typ `POINTER_AUTOMATIC`, was bedeutet, dass dieses Objekt automatisch vom Medium des `mql5_programms` erzeugt wurde. Solche Objekte werden nach Verwendung auch automatisch entfernt.

### ENUM\_POINTER\_TYPE

| Konstante                      | Beschreibung                                                            |
|--------------------------------|-------------------------------------------------------------------------|
| <code>POINTER_INVALID</code>   | Unkorrekter Anzeiger                                                    |
| <code>POINTER_DYNAMIC</code>   | Anzeiger des Objektes, erzeugt von der Anweisung <a href="#">new()</a>  |
| <code>POINTER_AUTOMATIC</code> | Anzeiger jedes Objektes, erzeugt automatisch (ohne <code>new()</code> ) |

Sehen Sie auch

[Ausführungsfehler](#), [Anweisung der Objektentfernung delete](#), [CheckPointer\(\)](#)



## Andere Konstanten

Konstante CLR\_NONE wird verwendet, um Farbefehlen anzudeuten, d.h. [graphisches Objekt](#) oder [graphische Serie](#) des Indikatoren wird nicht dargestellt. Diese Konstante gehört zur Liste der Benennungen von [Web-Farben](#) nicht, kann aber überall verwendet werden, wo Farbenargumente angefordert werden.

Konstante INVALID\_HANDLE kann bei der Prüfung von Dateihandles (s. die Funktionen [FileOpen\(\)](#) und [FileFindFirst\(\)](#)) verwendet werden.

| Konstante       | Beschreibung                                                                                         | Wert                                              |
|-----------------|------------------------------------------------------------------------------------------------------|---------------------------------------------------|
| CHARTS_MAX      | Maximale mögliche Zahl der gleichzeitig offenen Charts im Terminal                                   | 100                                               |
| clrNONE         | Abwesenheit der Farbe                                                                                | -1                                                |
| EMPTY_VALUE     | Leerwert im Indikatorpuffer                                                                          | DBL_MAX                                           |
| INVALID_HANDLE  | unkorrektes handle                                                                                   | -1                                                |
| IS_DEBUG_MODE   | Flagge der Arbeit des mq5-Programm in Debugging-Mode                                                 | true in Debugging-Mode, anderenfalls false        |
| IS_PROFILE_MODE | Flagge der Arbeit des mq5-Programms im Modus von Code Profiling                                      | im Profiling-Modus ist nicht gleich null, sonst 0 |
| NULL            | Null des jeden Typs                                                                                  | 0                                                 |
| WHOLE_ARRAY     | Bedeutet die Anzahl von Elementen, die bis zum Feldende bleiben, bis das ganze Feld verarbeitet wird | -1                                                |
| WRONG_VALUE     | Konstante kann implizit <a href="#">reduziert werden</a> zum Typ jeder <a href="#">Aufzählung</a> .  | -1                                                |

Konstante EMPTY\_VALUE entspricht gewöhnlich den Werten des Indikators, die auf dem Chart nicht gezeigt werden. ZB. Für den eingebauten Indikator Standard Deviation mit der Periode wird die Linie für die ersten 19 Bars in der Geschichte nicht ausgegeben. Wenn Sie handle dieses Indikators durch die Funktion [iStdDev\(\)](#) erzeugen und die Indikatorwerte für diese Bars in Feld durch [CopyBuffer\(\)](#) kopieren, werden diese Werte EMPTY\_VALUE gleich sein.

Man kann selbstständig seinen eigenen Leerwert des Indikators im [Benutzerindikator](#) spezifizieren, wobei der Indikator auf dem Chart nicht gezeigt zu werden braucht. Verwenden Sie dafür die Funktion [PlotIndexSetDouble\(\)](#) mit dem Modifikator [PLOT\\_EMPTY\\_VALUE](#).

Konstante [NULL](#) kann einer Variable jedes einfachen Typs oder Anzeiger des Objektes oder der Struktur oder der Klasse zugeordnet werden. Die Zuordnung von NULL einer Zeilenvariable bedeutet die vollständige Deinitialisierung dieser Variable.

Konstante WRONG\_VALUE ist für die Fäellen bestimmt, wenn es erforderlich ist, den Wert der

Aufzählung rückzugeben und das muss ein unkorrekter Wert sein muss. ZB man muss angeben, dass dieser Wert ein Wert dieser Aufzählung ist. Als Beispiel kann man eine gewisse Funktion `CheckLineStyle()` nehmen, die Linienstil für Objekt rückgibt, das mit dem Namen angegeben wird. Wenn bei der Stilanforderung durch die Funktion `ObjectGetInteger()` das Ergebnis `true` sein wird, kehrt ein Wert der Aufzählung `ENUM_LINE_STYLE` zurück, anderenfalls kehrt `WRONG_VALUE` zurück.

```
void OnStart()
{
    if (CheckLineStyle("MyChartObject")==WRONG_VALUE)
        printf("Error line style getting.");
}
//+-----+
//| Kehrt Stil der Linie für Objekt, das mit dem Namen angegeben wird |
//+-----+
ENUM_LINE_STYLE CheckLineStyle(string name)
{
    long style;
//---
    if (ObjectGetInteger(0,name,OBJPROP_STYLE,0,style))
        return ((ENUM_LINE_STYLE)style);
    else
        return (WRONG_VALUE);
}
```

Konstante `WHOLE_ARRAY` ist für die Funktionen bestimmt, die die Andeutung der Anzahl der Elementen in verarbeiteten Felder fordern:

- [ArrayCopy\(\)](#);
- [ArrayMinimum\(\)](#);
- [ArrayMaximum\(\)](#);
- [FileReadArray\(\)](#);
- [FileWriteArray\(\)](#).

Wenn man spezifizieren muss, dass alle Feldwerte von der angegebenen Position bis zum Ende verarbeitet werden muessen, reicht es, den Wert `WHOLE_ARRAY`. anzugeben.

Konstante `IS_PROFILE_MODE` erlaubt die Arbeit des Programms für die korrekte Datenerfassung im Profiling-Modus zu ändern. Profiling erlaubt die Ausführungszeit von einzelnen Fragmente des Programms (in der Regel eine Funktion) zu messen, und auch die Anzahl solcher Aufrufe zu berechnen. Um die korrekte Information über die Ausführungszeit im Profiling-Modus zu erhalten, kann man die Aufrufe der Funktion `Sleep()` wie folgt deaktivieren:

```
//--- Sleep kann das Ergebnis von Profiling
if(!IS_PROFILE_MODE) Sleep(100) stark beeinflussen(verzerren); // verbieten wir den Au
```

Wert der Konstante `IS_PROFILE_MODE` wird vom Compiler bei der Compilierung angegeben, und in gewöhnlichen Modus gleich Null gestellt. Beim Programmstart im Profiling-Modu wird eine spezielle Compilierung durchgeführt, und in diesem Fall statt `IS_PROFILE_MODE` wird einen von null verschiedenen Wert ersetzt.

Das Verwenden der Konstante `IS_DEBUG_MODE` ist für die leichte Veränderung der Arbeit `mql5`-Programms in Debuggin-Mode nuetzlich. ZB. in Debugging-Mode kann es erforderlich sein, zusaetzliche Debugging Information in Log des Terminals auszugeben oder zusaetzliche graphische Objekte auf dem Chart zu erzeugen.

Das unten angeführte Beispiel erzeugt das Objekt Label und gibt seine Beschreibung und Farbe abhängig vom Mode des Scriptablaufs. Für Ablauf des Scripts in Debugging-Mode von MetaEditor, drücken Sie die Taste F5. Wenn Sie wollen, Script vom Browserwindow im Terminal ablaufen lassen, wird Farbe und Objekttext von Label anders sein.

#### Beispiel:

```
//+-----+
//|                                     Check_DEBUG_MODE.mq5 |
//|                                     Copyright © 2009, MetaQuotes Software Corp. |
//|                                     http://www.metaquotes.net |
//+-----+
#property copyright "Copyright © 2009, MetaQuotes Software Corp."
#property link      "http://www.metaquotes.net"
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//---
    string label_name="invisible_label";
    if(ObjectFind(0,label_name)<0)
    {
        Print("Object ",label_name," not found. Error code = ",GetLastError());
        //--- erzeugen wir das Objekt Label
        ObjectCreate(0,label_name,OBJ_LABEL,0,0,0);
        //--- stellen wir die Koordinate X ein
        ObjectSetInteger(0,label_name,OBJPROP_XDISTANCE,200);
        //--- stellen wir die Koordinate Y ein
        ObjectSetInteger(0,label_name,OBJPROP_YDISTANCE,300);
        ResetLastError();
        if(IS_DEBUG_MODE) // Debugging-Mode
        {
            //--- Nachricht über Mode des Scriptablaufs zeigen
            ObjectSetString(0,label_name,OBJPROP_TEXT,"DEBUG MODE");
            //--- geben wir rote Farbe des Textes vor
            if(!ObjectSetInteger(0,label_name,OBJPROP_COLOR,clrRed))
                Print("Nicht in der Lage, Farbe einzustellen. Fehler ",GetLastError());
        }
        else // Arbeitsmode
            ObjectSetString(0,label_name,OBJPROP_TEXT,"RELEASE MODE");
        //--- stellen wir unsichtbare Textfarbe vor
        if(!ObjectSetInteger(0,label_name,OBJPROP_COLOR,CLR_NONE))
            Print("Misskungen, Farbe einzustellen. Fehler ",GetLastError());
    }
    ChartRedraw();
    DebugBreak(); // hier wird Unterbrechung erfolgen, wenn wir in Debugging-Mode
}
}
```

## Методы шифрования данных

Для указания метода преобразования данных (шифрование и расчет хешей) в функциях [CryptEncode\(\)](#) и [CryptDecode\(\)](#) используется перечисление `ENUM_CRYPT_METHOD`.

### ENUM\_CRYPT\_METHOD

| Константа         | Описание                                   |
|-------------------|--------------------------------------------|
| CRYPT_BASE64      | Шифрование BASE64 (перекодировка)          |
| CRYPT_AES128      | Шифрование AES с ключом 128 бит (16 байт)  |
| CRYPT_AES256      | Шифрование AES с ключом 256 бит (32 байта) |
| CRYPT_DES         | Шифрование DES с ключом 56 бит (7 байт)    |
| CRYPT_HASH_SHA1   | Расчёт HASH SHA1                           |
| CRYPT_HASH_SHA256 | Расчёт HASH SHA256                         |
| CRYPT_HASH_MD5    | Расчёт HASH MD5                            |
| CRYPT_ARCH_ZIP    | ZIP архивирование                          |

Sehen Sie auch

[DebugBreak](#), [Information über das ausgeführte MQL5-Programm](#), [CryptEncode\(\)](#), [CryptDecode\(\)](#)

## Datenstrukturen

In MQL5 gibt es 8 vorbestimmte Strukturen, die für Speicherung und Übertragung der Sonderinformation bestimmt sind:

- MqlDateTime ist bestimmt für die Darstellung des Datums und der Zeit;
- MqlParam ermöglicht Eingabeparameter zu übertragen bei der Erzeugung von handle des Anzeigers durch die Funktion IndicatorCreate();
- MqlRates bietet Information über historische Daten, sie enthalten Preis, Volumen und Erweiterung;
- MqlBookInfo für Erhaltung der Information von DOM (Fenster der Quotationen) ;
- MqlTradeRequest ist verwendet für Erzeugung der Handelsanforderung bei der Durchführung von Handelsoperationen;
- MqlTradeResult hat die Antwort des Handelsservers auf die Handelsanforderung, gesendet durch die Funktion OrderSend();
- MqlTradeTransaction enthält die Beschreibung einer Handelstransaktion;
- MqlTick ist verwendet, um schnelle gefragteste Information über die laufenden Preise zu bekommen.

## MqlDateTime

In der Datumstruktur gibt es acht Felder des Typs [int](#).

```
struct MqlDateTime
{
    int year;           // Jahr
    int mon;            // Monat
    int day;            // Tag
    int hour;           // Stunde
    int min;            // Minuten
    int sec;            // Sekunden
    int day_of_week;    // Wochentag (0-Sonntag, 1-Montag)
    int day_of_year;    // Ordnungszahl des Jahres (1. Februar - ist 32.Tag im Jahr)
};
```

### Bemerkung

Ordnungszahl des Jahres `day_of_year` im Schaltjahr, angefangen mit Maerz, wird sich von der Ordnungszahl des entsprechenden Tages im Nichtschaltjahr unterscheiden.

### Beispiel:

```
void OnStart()
{
    //---
    datetime date1=D'2008.03.01';
    datetime date2=D'2009.03.01';

    MqlDateTime str1,str2;
    TimeToStruct(date1,str1);
    TimeToStruct(date2,str2);
    printf("%02d.%02d.%4d, day of year = %d",str1.day,str1.mon,
        str1.year,str1.day_of_year);
    printf("%02d.%02d.%4d, day of year = %d",str2.day,str2.mon,
        str2.year,str2.day_of_year);
};
/* Ergebnis
01.03.2008, day of year = 60
01.03.2009, day of year = 59
*/
```

### Sehen Siw auch

[TimeToStruct](#), [Strukturen und Klassen](#)

## Struktur der Eingabeparameter des Indikators (MqlParam)

Struktur MqlParam ist speziell für Übertragung der [Eingabeparameter](#) ausgearbeitet bei der Erzeugung von handle des [technischen Indikators](#) durch die Funktion [IndicatorCreate\(\)](#).

```
struct MqlParam
{
    ENUM_DATATYPE    type;                // Typ des Eingabeparameters, Wert der Aufzählung ENUM
    long              integer_value;       // Feld für Speicherung des ganzzahligen Wertes
    double            double_value;        // Feld für Speicherung des Wertes double oder float
    string            string_value;        // Feld für Speicherung des Wertes des Zeilentyps
};
```

Alle Eingabeparameter des Indikators werden als Feld des Typs MqlParam übertragen, Feld *type* jedes Elementes dieses Feldes deutet auf den Datentyp, die von diesem Element übertragen werden. Die Werte der Indikatorparameter müssen vorläufig in entsprechende Felder jedes Elementes stellen in *integer\_value*, in *double\_value* oder in *string\_value*) abhängig davon welcher Wert der Aufzählung [ENUM\\_DATATYPE](#) im Feld *type* gibt.

Wenn der Wert [IndicatorCreate\(\)](#) als Indikatortyp vom dritten Parameter durch die Funktion IND\_CUSTOM übertragen wird, muss das erste Element des Feldes der Eingabeparameter Feld *type* mit dem Wert TYPE\_STRING aus der Aufzählung [ENUM\\_DATATYPE](#) haben und Feld *string\_value* muss den Namen des [Benutzerindikators](#) enthalten.

## MqlRates

Die Struktur bewahrt die Information über die Preise, Volumen und Spread auf.

```
struct MqlRates
{
    datetime time;           // Anfangszeit der Periode
    double open;             // Eröffnungspreis
    double high;             // der hoechste Preis der Periode
    double low;              // der niedrigste Preis der Periode
    double close;            // Schlusspreis
    long tick_volume;        // Tickvolumen
    int spread;              // Spread
    long real_volume;        // Lagervolumen
};
```

### Beispiel:

```
void OnStart()
{
    MqlRates rates[];
    int copied=CopyRates(NULL,0,0,100,rates);
    if(copied<=0)
        Print("Fehler des Kopieren von Preisdaten ",GetLastError());
    else Print("Kopiert ",ArraySize(rates)," Bars");
};
```

### Sehen Sie auch

[CopyRates](#), [Zugang zu Timeserien](#)



## MqlBookInfo

In der Struktur gibt es Information über DOM.

```
struct MqlBookInfo
{
    ENUM_BOOK_TYPE    type;        // Typ der Anforderung aus der Aufzählung ENUM\_BOOK\_TYPE
    double             price;       // Preis
    long               volume;     // Volumen
};
```

### Bemerkung

MqlBookInfo ist eine vordefinierte Struktur, so braucht sie keine Deklaration und Beschreibung. Um die Struktur zu benutzen, nur deklarieren Sie eine Variable dieses Typs.

DOM ist nur für finanzielle Symbole zugaenglich.

### Beispiel:

```
MqlBookInfo priceArray[];
bool getBook=MarketBookGet(NULL,priceArray);
if(getBook)
{
    int size=ArraySize(priceArray);
    Print("MarketBookInfo für ",Symbol());
};
else
{
    Print("Missslungen, DOM des Symbols zu erhlaten ",Symbol());
};
```

### Sehen Sie auch

[MarketBookAdd](#), [MarketBookRelease](#), [MarketBookGet](#), [Handelsordern in DOM](#), [Datentypen](#)

## Struktur der Handelsanforderung (MqlTradeRequest)

Interaktion von Kundenterminal und Handelsserver für Durchführung für Operation der Orderstellung wird mittels Handelsanforderungen durchgeführt. Anforderung ist durch die spezielle vorbestimmte Struktur `MqlTradeRequest` dargestellt, die alle erforderlichen Felder für Abschluss der Handelsorder enthält. Ergebnis der Verarbeitung von Anforderung ist durch die Struktur `MqlTradeResult` repräsentiert.

```
struct MqlTradeRequest
{
    ENUM_TRADE_REQUEST_ACTIONS    action;           // Operationstyp
    ulong                        magic;               // Expert Advisor ID (Identifikator magic number)
    ulong                        order;               // Orderticket
    string                       symbol;              // Name des Handelssymbols
    double                       volume;              // angefordertes Dealvolumen in Lots
    double                       price;               // Preis
    double                       stoplimit;           // Orderlevel StopLimit
    double                       sl;                  // Oderlevel Stop Loss
    double                       tp;                  // Orderlevel Take Profit
    ulong                        deviation;           // maximal moegliche Abweichung vom angeforderte
    ENUM_ORDER_TYPE              type;                // Ordertyp
    ENUM_ORDER_TYPE_FILLING      type_filling;        // Durchführungstyp der Order
    ENUM_ORDER_TYPE_TIME         type_time;          // Wirkzeittyp der Order
    datetime                     expiration;          // Fristablauf der Order (für Ordnern des Typs ORD
    string                       comment;             // Oprederkommentar
};
```

### Beschreibung der Felder

| Feld   | Beschreibung                                                                                                                                                                                    |
|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| action | Typ der Handelsoperation. Wert kann einer der Aufzählungswerte <u>ENUM_TRADE_REQUEST_ACTIONS</u> sein                                                                                           |
| magic  | Identifikator des Experten. Ermoeeglicht, analytische Verarbeitung der Handelsordern zu gestalten. Jeder Experte kann seinen unikalnen Identifikator beim Senden der Handelsanforderung stellen |
| order  | Ticket der Order. Für Modifikation der Warteordern erforderlich.                                                                                                                                |
| symbol | Symbolname der Order. Nicht erforderlich bei Modifikation der Ordnern und Schliessen der Positionen.                                                                                            |
| volume | Angefordertes Dealvolumen in Lots. Realwert des Volumens wird vom <u>Durchführungstyp der Order</u> abhängen                                                                                    |
| price  | Preis, bei dem Order ausgeführt werden muss, für Markttordern mit dem Durchführungstyp "Market Execution" ( <u>SYMBOL_TRADE_EXECUTION_MARK</u>                                                  |

|              |                                                                                                                                                                                       |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|              | <a href="#">ET</a> ), die den Typ <a href="#">TRADE_ACTION_DEAL</a> haben, ist es nicht erforderlich, Preis anzugeben                                                                 |
| stoplimit    | Preis, bei dem Warteorder Limit gestellt wird, wenn der Preis den Wert price reicht( diese Bedingung ist obligatorisch). Bis dahin wird Warteorder ins Handelssystem nicht ausgegeben |
| sl           | Stop Loss des Preises bei unguenstiger Bewegung des Preises                                                                                                                           |
| tp           | Take Profit des Preises bei guenstiger Bewegung des Preises                                                                                                                           |
| deviation    | Maximal moefliche Abweichung vom angeforderten Preis vorgegeben in <a href="#">Punkten</a>                                                                                            |
| type         | Ordertyp. Wert kann einer der Aufzählungswerte <a href="#">ENUM_ORDER_TYPE</a> sein                                                                                                   |
| type_filling | Durchführungstyp der Order. Kann einer der Aufzählungswerte <a href="#">ENUM_ORDER_TYPE_FILLING</a> sein                                                                              |
| type_time    | Ablauftyp der Order. Kann einer der Aufzählungswerte <a href="#">ENUM_ORDER_TYPE_TIME</a> sein                                                                                        |
| expiration   | Ablaufzeit der Warteordern (für Ordnern des Typs <a href="#">ORDER_TIME_SPECIFIED</a> )                                                                                               |
| comment      | Kommentar zur Order                                                                                                                                                                   |

für Senden der Order [Handelsoperationen](#) durchzuführen, muss die Funktion [OrderSend\(\)](#) verwendet werden. Für jede Handelsoperation muessen obligatorische Felde angegeben werden und können optionale Felder ausgefüllt werden. Insgesamt werden 7 Varianten des Sendens der Handelsanforderung vorausgesehen:

### Request Execution

Handelsorder, Positionen im Request Execution Mode zu oeffnen (Handelsmode bei der Anforderung der laufenden Preise). Man muss 9 Felder spezifizieren:

- action
- symbol
- volume
- price
- sl
- tp
- deviation
- type
- type\_filling

Man kann auch Feldwerte magic und comment vorgeben.

### Instant Execution

Handelsorder, Positionen im Instant Execution Mode zu oeffnen (Handel mit Flowpreisen). Man muss 9 Felder spezifizieren:

- action
- symbol
- volume
- price
- sl
- tp
- deviation
- type
- type\_filling

Man kann auch Feldwerte magic und comment vorgeben.

### Market Execution

Handelsorder, Positionen im Market Execution Mode zu oeffnen (Durchführung der Handelsorder im Mark). Man muss 5 Felder spezifizieren:

- action
- symbol
- volume
- type
- type\_filling

Man kann auch Feldwerte magic und comment vorgeben.

### Exchange Execution

Handelsorder, Positionen im Exchange Execution Modus zu öffnen (Börse-Modus der Ausführung von Handelsordern). Man muss 5 Felder angeben:

- action
- symbol
- volume
- type
- type\_filling

Man kann auch Feldwerte magic und comment vorgeben.

### SL & TP Modification

Handelsorder, Levels StopLoss und/oder TakeProfit zu modifizieren. Man muss 4 Felder angeben:

- action
- symbol
- sl
- tp

### Pending Order

Handelsorder, eine Handelsorder einzustellen. Man muss 11 Felder angeben:

- action

- symbol
- volume
- price
- stoplimit
- sl
- tp
- type
- type\_filling
- type\_time
- expiration

Man kann auch Feldwerte magic und comment vorgeben.

### Modify Pending Order

Handelsorder, Preislevels der Warteorder zu modifizieren. Man muss 7 Felder spezifizieren:

- action
- order
- price
- sl
- tp
- type\_time
- expiration

### Delete Pending Order

Handelsorder, eine Warteorder zu entfernen. Man muss 2 Felder spezifizieren:

- action
- order

### Sehen Sie auch

[Strukturen und Klassen](#), [Handelsfunktionen](#), [Ordereigenschaften](#)

## Struktur der Ergebnisse der Prüfung der Handelsanforderung (MqlTradeCheckResult)

Bevor Sie dem Handelsserver [Anforderung](#) auf eine [Handelsoperation](#) senden, muss man es prüfen. Die Prüfung wird mittels der Funktion [OrderCheck\(\)](#) durchgeführt, der die geprüfte Anforderung und die Variable des Typs Strukturen MqlTradeCheckResult übertragen werden. In diese Variable wird das Ergebnis der Prüfung geschrieben.

```
struct MqlTradeCheckResult
{
    uint      retcode;           // Antwortkode
    double    balance;          // Bilanz nach Dealabschluss
    double    equity;           // Equity nach Dealabschluss
    double    margin;           // Margenanforderungen
    double    margin_free;      // Freie Marge
    double    margin_level;     // Level der Marge
    string    comment;          // Kommentar zum Antwortkode (Beschreibung des Fehlers)
};
```

### Beschreibung der Felder

| Feld         | Beschreibung                                                                                   |
|--------------|------------------------------------------------------------------------------------------------|
| retcode      | <a href="#">Rückgabekode</a>                                                                   |
| balance      | Wert der Bilanz nach Ausführung der Handelsoperation                                           |
| equity       | Wert der Eigenmittel nach Ausführung der Handelsoperation                                      |
| profit       | Wert des fließenden Gewinns nach Ausführung der Handelsoperation                               |
| margin       | Grösse der Marge, die für die erforderliche Handelsoperation notwendig ist                     |
| margin_free  | Grösse der freien Eigenmittel, die nach Ausführung der erforderlichen Handelsoperation bleiben |
| margin_level | Levekl der Marge nach Ausführung der erforderlichen Handelsoperation                           |
| comment      | Kommentar zum Antwortkode, Beschreibung des Fehlers                                            |

### Sehen Sie auch

[Struktur der Handelsanforderung](#), [Struktur für Erhaltung der laufenden Preise](#), [OrderSend](#), [OrderCheck](#)

# Struktur des Ergebnisses der Handelsanforderung (MqlTradeResult)

Als Antwort auf [Handelsanforderung](#), gibt das Handelsserver Daten über Bearbeitungsergebnis der Handelsanforderung als eine vorbestimmte Sonderstruktur MqlTradeResult zurück.

```
struct MqlTradeResult
{
    uint      retcode;           // Kode des Operationsergebnisses
    ulong     deal;             // Ticket des Deals, wenn es abgeschlossen ist
    ulong     order;            // Ticket der Order, wenn sie gestellt ist
    double    volume;           // Dealvolumen, bestaetigt vom Broker
    double    price;            // Dealpreis, bestaetigt vom Broker
    double    bid;              // Laufender Bidpreis (Requote Preis)
    double    ask;              // Laufender Askpreis (Requote Preis)
    string     comment;          // Kommentar des Brokers zur Opearion (als Default durch Beschreibung
    uint      request_id;        // Identifikator der Anforderung, der vom Terminal beim Senden eingese
};
```

## Beschreibung der Felder

| Feld       | Beschreibung                                                                                                                                                              |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| retcode    | <a href="#">Rückkehrkode</a> des Handelsservers                                                                                                                           |
| deal       | Ticket von <a href="#">Deal</a> , wenn es abgeschlossen ist. Zugaenglich bei der Handelsoperation <a href="#">TRADE_ACTION_DEAL</a>                                       |
| order      | Ticket der <a href="#">Order</a> , wenn sie gestellt ist. Gemeldet bei der Handelsoperation <a href="#">TRADE_ACTION_PENDING</a>                                          |
| volume     | Dealvolumen, bestaetigt vom Broker. Haengt vom <a href="#">Typ der Orderausfuellung</a>                                                                                   |
| price      | Preis im Deal, bestaetigt vom Broker. Haengt vom Feld <i>deviation</i> in der <a href="#">Handelsanforderung</a> und/oder vom Typ der <a href="#">Handelsoperation</a> ab |
| bid        | Laufender Bidpreis (Requote Preis)                                                                                                                                        |
| ask        | Laufender Askpreis (Requote Preis)                                                                                                                                        |
| comment    | Kommentar des Brokers zur Operation (als Default durch Beschreibung ausgefuellt Rückkehrcode des Handels Servers)                                                         |
| request_id | Identifikator der Anforderung, der vom Terminal beim Senden eingesetzt wird                                                                                               |

Ergebnis der Handelsoperation kehrt zur Variable des Typs MqlTradeResult zurück, die vom zweiten Parameter in die Funktion [OrderSend\(\)](#) für Durchführung der [Handelsoperationen](#) übertragen wird .

Das Terminal schreibt den Identifikator der [Anforderung](#) in das Feld request\_id, wenn er an den Handel-Server durch Funktionen [OrdersSend\(\)](#) und [OrderSendAsync\(\)](#) gesendet wird. Vom Handel-Server erhält das Terminal Nachrichten über die ausgeführten Handelstransaktionen und übergibt sie für Bearbeitung in die [OnTradeTransaction\(\)](#) Funktion, die als die Parameter enthält:

- Beschreibung der Handelstransaktion in der Struktur [MqlTradeTransaction](#);
- >Beschreibung der [Handelsanforderung](#), der aus der Funktion OrderSend() oder OrderSendAsync() gesendet ist. Identifikator der Anforderung wird vom Terminal an den Handle-Server gesendet, und die Anforderung und seinen request\_id werden im Terminalspeicher aufbewahrt;
- das Ergebnis der Ausführung der Handelsanforderung als die Struktur MqlTradeResult, in dem das Feld request\_id den Identifikator dieser Anforderung enthält.

Funktion OnTradeTransaction() bekommt drei Eingabeparameter, aber die beiden letzten Parameter sollten nur für die Handelstransaktionen mit dem Typ [TRADE\\_TRANSACTION\\_REQUEST](#) analysiert werden. In allen anderen Fällen werden die Daten über die Handelsanforderung und das Ergebnis seiner Ausführung nicht ausgefüllt. Das Beispiel der Analyse der Parameter ist im Abschnitt [Struktur der Handelstransaktion](#) gegeben.

Die Einstellung des Identifikators request\_id vom Terminal für die Handelsanforderung dient zur Arbeit mit der asynchronen Funktion OrderSendAsync() beim Senden an den Server. Dieser Identifikator erlaubt die ausgeführte Aktion (Aufruf der Funktion OrderSend oder OrderSendAsync) mit dem in die [OnTradeTransaction\(\)](#) übergebenen Ergebnis dieser Aktion zu verbinden.

**Beispiel:**



```

//+-----+
//| Senden der Handelsanforderung mit Ergebnisverarbeitung |
//+-----+
bool MyOrderSend(MqlTradeRequest request,MqlTradeResult result)
{
//--- stellen wir Kode des letzten Fehlers auf Null
ResetLastError();
//--- senden wir Anforderung
bool success=OrderSend(request,result);
//--- wenn das Ergebnis erfolglos ist - versuchen wir zu erfahren warum
if(!success)
{
int answer=result.retcode;
Print("TradeLog:Trade request failed. Error = ",GetLastError());
switch(answer)
{
//--- Requote
case 10004:
{
Print("TRADE_RETCODE_REQUOTE");
Print("request.price = ",request.price," result.ask = ",
result.ask," result.bid = ",result.bid);
break;
}
//--- Order ist vom Server nicht akzeptiert
case 10006:
{
Print("TRADE_RETCODE_REJECT");
Print("request.price = ",request.price," result.ask = ",
result.ask," result.bid = ",result.bid);
break;
}
//--- Falscher Preis
case 10015:
{
Print("TRADE_RETCODE_INVALID_PRICE");
Print("request.price = ",request.price," result.ask = ",
result.ask," result.bid = ",result.bid);
break;
}
//--- falscher SL und/oder TP
case 10016:
{
Print("TRADE_RETCODE_INVALID_STOPS");
Print("request.sl = ",request.sl," request.tp = ",request.tp);
Print("result.ask = ",result.ask," result.bid = ",result.bid);
break;
}
//--- unkorrektes Volumen
case 10014:
{
Print("TRADE_RETCODE_INVALID_VOLUME");
Print("request.volume = ",request.volume," result.volume = ",
result.volume);
break;
}
//--- nicht genug Geld für Handelsoperation
case 10019:
{
Print("TRADE_RETCODE_NO_MONEY");
Print("request.volume = ",request.volume," result.volume = ",

```

```
        result.volume,"    result.comment = ",result.comment);
    break;
}
//--- irgendwelcher anderer Grund, Kode der Serverantwort ist zu melden
default:
{
    Print("Other answer = ",answer);
}
}
//--- erfolgloses Ergebnis der Handelsanforderung durch Rückkehr false wird gemei
return(false);
}
//--- OrderSend() gibt true zurück - wiederholen wir die Antwort
return(true);
}
```

## Struktur der Handelstransaktion (MqlTradeTransaction)

Als Ergebnis der Durchführung von bestimmten Aktionen auf dem Handelskonto, ändert sich der Kontostand. Zu solchen Aktionen gehören:

- Senden der Handelsanfrage aus einer beliebigen MQL5-Anwendung in dem Kundenterminal mittels der Funktionen [OrderSend](#) und [OrderSendAsync](#) und ihre nachfolgende Ausführung;
- Senden der Handelsanfrage über die grafische Oberfläche des Terminals und ihre nachfolgende Ausführung;
- Auslösen von aufgeschobenen Ordnern und Stop-Ordnern auf dem Server;
- Ausführen von Operationen auf der Seite der Handel-Server.

Als Ergebnis dieser Aktionen werden die folgenden Handelstransaktionen durchgeführt:

- Die Verarbeitung der Handelsanfrage;
- Änderung der offenen Aufträge;
- Änderung der Ordergeschichte;
- Änderung der Dealgeschichte;
- Änderung der Position.

Zum Beispiel, beim Senden Marktkauforder, wird Marktkauforder behandelt und eine entsprechende Marktkauforder für das Konto erstellt. Ist die Order ausgeführt, wird sie aus der Auftragsliste entfernt und zur Ordergeschichte hinzugefügt. Dann wird der entsprechende Deal zur Geschichte hinzugefügt und eine neue Position wird erstellt. Alle diese Aktionen sind Handelstransaktionen.

Um die auf das Konto angewendeten Handelstransaktionen zu erhalten, ist der spezielle Handler [OnTradeTransaction](#) im MQL5 vorgesehen. In den ersten Parameter wird die Struktur MqlTradeTransaction übergeben, die Handelstransaktionen beschreibt.

```
struct MqlTradeTransaction
{
    ulong                deal;           // Ticket des Deals
    ulong                order;          // Ticket der Order
    string               symbol;         // Name des Handelsinstrumentes
    ENUM_TRADE_TRANSACTION_TYPE type;    // Typ der Handelstransaktion
    ENUM_ORDER_TYPE       order_type;    // Typ der Order
    ENUM_ORDER_STATE      order_state;   // Zustand der Order
    ENUM_DEAL_TYPE        deal_type;     // Typ des Deals
    ENUM_ORDER_TYPE_TIME  time_type;     // Typ der Order nach der Zeit der Aktion
    datetime              time_expiration; // Ablaufzeit der Order
    double                price;          // Preis
    double                price_trigger;  // Aktivierungspreis von Stop-Limit-Order
    double                price_sl;       // Stop Loss Ebene
    double                price_tp;       // Take Profit Ebene
    double                volume;         // Volumen in Lots
};
```

### Felder Beschreibung

| Feld  | Beschreibung     |
|-------|------------------|
| deal  | Ticket des Deals |
| order | Ticket der Order |

|                 |                                                                                                                                                  |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| symbol          | Name des Handelsinstrumente, nach dem die Transaktion durchgeführt wurde.                                                                        |
| type            | Typ der Handelstransaktion. Der Wert kann einer der Aufzählungswerte <a href="#">ENUM_TRADE_TRANSACTION_TYPE</a> sein.                           |
| order_type      | Typ der Handelsorder. Der Wert kann einer der Aufzählungswerte <a href="#">ENUM_ORDER_TYPE</a> sein.                                             |
| order_state     | Zustand der Handelsorder. Der Wert kann einer der Aufzählungswerte <a href="#">ENUM_ORDER_STATE</a> sein.                                        |
| deal_type       | Typ des Deals. Der Wert kann einer der Aufzählungswerte <a href="#">ENUM_DEAL_TYPE</a> sein.                                                     |
| type_time       | Typ der Order nach Ablauf. Der Wert kann einer der Aufzählungswerte <a href="#">ENUM_ORDER_TYPE_TIME</a> sein.                                   |
| time_expiration | Ablaufzeit der aufgeschobenen Order (für die Order von Typ <a href="#">ORDER_TIME_SPECIFIED</a> und <a href="#">ORDER_TIME_SPECIFIED_DAY</a> ).  |
| price           | Preis. Je nach Typ der Handelstransaktion kann es ein Preis von einer Order, eines Deals oder einer Position sein.                               |
| price_trigger   | Stop-Kurs (Aktivierungspreis) von Stop-Limit-Order ( <a href="#">ORDER_TYPE_BUY_STOP_LIMIT</a> und <a href="#">ORDER_TYPE_SELL_STOP_LIMIT</a> ). |
| price_sl        | Stop Loss Preis. Je nach Typ der Handelstransaktion kann es sich auf eine Order, einen Deal oder eine Position beziehen.                         |
| price_tp        | Take Profit Preis. Je nach Typ der Handelstransaktion kann es sich auf eine Order, einen Deal oder eine Position beziehen.                       |
| volume          | Volumen in Lots. Je nach Typ der Handelstransaktion kann es das aktuelle Volumen von einer Order, einem Deal oder einer Position zeigen.         |

Der bestimmende Parameter für die Analyse der empfangenen Transaktion ist sein Typ, der ins Feld **type** übergeben wird. Zum Beispiel, wenn eine Transaktion der Typ [TRADE\\_TRANSACTION\\_REQUEST](#) (das Ergebnis der Bearbeitung der Handelsanforderung vom Server ist erhalten) ist, hat die Struktur nur ein ausgefülltes Feld **type**. Die anderen Felder müssen nicht analysiert werden. In diesem Fall kann man zwei zusätzliche Parameter **request** und **result** analysieren, die an den Handler `OnTradeTransaction()` übergeben werden, wie im Beispiel unten gezeigt.

Wenn Sie den Typ der Handelsoperation wissen, können Sie über eine Analyse des aktuellen Zustandes von Ordnern, Positionen und Deals auf dem Handelskonto entscheiden. Es ist zu beachten, dass eine

Handelsanforderung, die aus dem Terminal an den Server gesendet wird, mehrere Handelstransaktionen auslösen kann, deren Reihenfolge des Eingangs ins Terminal nicht gewährleistet wird.

Die Struktur `MqlTradeTransaction` wird auf unterschiedlicher Weise je nach Typ der Handelstransaktion ([ENUM\\_TRADE\\_TRANSACTION\\_TYPE](#)) ausgefüllt:

#### TRADE\_TRANSACTION\_ORDER\_\* und TRADE\_TRANSACTION\_HISTORY\_\*

Für die Handelstransaktionen betreffend die Bearbeitung der offenen Ordern (`TRADE_TRANSACTION_ORDER_ADD`, `TRADE_TRANSACTION_ORDER_UPDATE` und `TRADE_TRANSACTION_ORDER_DELETE`) und der Ordergeschichte (`TRADE_TRANSACTION_HISTORY_ADD`, `TRADE_TRANSACTION_HISTORY_UPDATE`, `TRADE_TRANSACTION_HISTORY_DELETE`), werden in der Struktur `MqlTradeTransaction` die folgenden Felder ausgefüllt:

- `order` - Ticket der Order;
- `symbol` - Name eines Finanzinstrumentes in der Order;
- `type` - Typ der Handelstransaktion;
- `order_type` - Typ der Order;
- `orders_state` - aktueller Zustand der Order;
- `time_type` - Typ des Ablaufs der Order;
- `time_expiration` - Ablaufzeit der Order (für die Ordern mit dem Typ des Ablaufs von [ORDER\\_TIME\\_SPECIFIED](#) und [ORDER\\_TIME\\_SPECIFIED\\_DAY](#));
- `price` - Preis in der Order, der von einem Kunden angegeben ist;
- `price_trigger` - Stop-Kurs des Auslösens von Stop-Limit-Order (nur für [ORDER\\_TYPE\\_BUY\\_STOP\\_LIMIT](#) und [ORDER\\_TYPE\\_SELL\\_STOP\\_LIMIT](#));
- `price_sl` - Stop Loss der Order (wird ausgefüllt, wenn in der Order angegeben ist);
- `price_tp` - Take Profit Preis der Order (wird ausgefüllt, wenn in der Order angegeben ist);
- `volume` - aktuelles Volumen der Order (nicht ausgeführt). Anfangsvolumen der Order kann man in der Ordergeschichte mit Hilfe der Funktion [HistoryOrders\\*](#) finden.

#### TRADE\_TRANSACTION\_DEAL\_\*

Für die Handelstransaktionen betreffend die Bearbeitung des Deals (`TRADE_TRANSACTION_DEAL_ADD`, `TRADE_TRANSACTION_DEAL_UPDATE` und `TRADE_TRANSACTION_DEAL_DELETE`), werden in der Struktur `MqlTradeTransaction` die folgenden Felder ausgefüllt:

- `deal` - Ticket des Deals;
- `order` - Ticket der Order, auf dessen Grundlage der Deal abgeschlossen wurde;
- `symbol` - Name eines Finanzinstrumentes im Deal;
- `type` - Typ der Handelstransaktion;
- `deal_type` - Typ des Deals;
- `price` - Preis, zu dem einen Deal abgeschlossen wurde;
- `price_sl` - Stop Loss Preis (wird ausgefüllt, wenn in der Order angegeben ist, auf dessen Grundlage der Deal abgeschlossen wurde);
- `price_tp` - Take Profit Preis (wird ausgefüllt, wenn in der Order angegeben ist, auf dessen Grundlage der Deal abgeschlossen wurde);
- `volume` - Volumen des Deals in Lots.

#### TRADE\_TRANSACTION\_POSITION

Für die Handelstransaktionen betreffend die Änderungen von Positionen, die mit der Ausführung des Deals nicht verbunden sind (TRADE\_TRANSACTION\_POSITION), werden in der Struktur MqlTradeTransaction die folgenden Felder ausgefüllt:

- symbol - Name eines Finanzinstrumentes der Position;
- type - Typ der Handelstransaktion;
- deal\_type - Typ der Position ([DEAL\\_TYPE\\_BUY](#) oder [DEAL\\_TYPE\\_SELL](#));
- price - gewogener Durchschnittskurs der Eröffnung von Position;
- price\_sl - Stop Loss Preis;
- price\_tp - Take Profit Preis;
- volume - Volumen der Position in Lots, wenn es geändert wurde.

Änderung der Position (Hinzufügen, Änderung oder Liquidation) als Ergebnis des Abschlusses des Deals zieht kein Erscheinen der Transaktion TRADE\_TRANSACTION\_POSITION nach sich.

#### TRADE\_TRANSACTION\_REQUEST

Nur ein Feld wird in der Struktur MqlTradeTransaction für die Handelstransaktionen ausgefüllt, die die Tatsache beschreiben, dass die Handelsanfrage vom Server bearbeitet ist und das Ergebnis seiner Bearbeitung empfangen ist (TRADE\_TRANSACTION\_REQUEST):

- type - Typ der Handelstransaktion;

Für Transaktionen dieses Typs muss nur type Feld (Typ der Handelstransaktion) analysiert werden. Für zusätzliche Informationen muss man die zweiten und dritten Parameter der Funktion [OnTradeTransaction](#) (request und result) analysieren.

Sehen Sie auch

[Typen der Handelstransaktionen](#), [OnTradeTransaction](#)

## Struktur für Erfassung der laufenden Preise (MqlTick)

Das ist die Struktur für Speicherung der letzten Preise des Symbols. Bestimmt für schnelles Erhalten der gefragtesten Information über laufende Preise.

```
struct MqlTick
{
    datetime    time;           // Zeit des letzten Updates der Preise
    double      bid;           // Laufender Preis Bid
    double      ask;           // Laufender Preis Ask
    double      last;          // Laufender Preis des letzten Deals (Last)
    ulong       volume;        // Volumen für laufenden Preis Last
};
```

Variable des Typs MqlTick ermöglicht mit einem Anruf der Funktion [SymbolInfoTick\(\)](#) Werte Ask, Bid, Last und Volume zu bekommen.

### Beispiel:

```
void OnTick()
{
    MqlTick last_tick;
    //---
    if(SymbolInfoTick(Symbol(),last_tick))
    {
        Print(last_tick.time,": Bid = ",last_tick.bid,
              " Ask = ",last_tick.ask," Volume = ",last_tick.volume);
    }
    else Print("SymbolInfoTick() failed, error = ",GetLastError());
    //---
};
```

### Sehen Sie auch

[Strukturen und Klassen](#)

## Kodes der Fehler und Warnungen

Im Abschnitt gibt es folgende Beschreibungen:

- [Rückkehrkodes des Handelsservers](#) - Analyse der Ergebnisse von [Handelsserver](#), gesendet durch die Funktion [OrderSend\(\)](#);
- [Compiler Warnungen](#) - Kodes der Warnmedungen, die bei der Compilierung erscheinen (sind keine Fehler);
- [Compilierungsfehler](#) - Kodes der Fehlernachrichten beim unerfolgreichen Versuch der Compilierung;
- [Laufzeitfehler](#) - Fehlerkodes bei der Durchführung des mql5-Programms, die durch die Funktion [GetLastError\(\)](#) erhalten werden können.



## Rückgabecodes des Handelsservers

Alle Befehlsanweisungen, Handelsoperationen durchzuführen, werden als Struktur der Handelsanforderung [MqlTradeRequest](#) mittels der Funktion [OrderSend\(\)](#) gesendet. Das Ergebnis der Erfüllung dieser Funktion wird in die Struktur [MqlTradeResult](#) eingetragen, deren Feld *retcode* Rückkehrkode des Handelsservers enthält.

| Kode  | Identifikator                    | Beschreibung                                                   |
|-------|----------------------------------|----------------------------------------------------------------|
| 10004 | TRADE_RETCODE_REQUOTE            | Requote                                                        |
| 10006 | TRADE_RETCODE_REJECT             | Anforderung ist abgelehnt                                      |
| 10007 | TRADE_RETCODE_CANCEL             | Anforderung ist vom Trader aufgehoben                          |
| 10008 | TRADE_RETCODE_PLACED             | Order ist platziert                                            |
| 10009 | TRADE_RETCODE_DONE               | Anforderung ist erfüllt                                        |
| 10010 | TRADE_RETCODE_DONE_PARTIAL       | Anforderung ist partiell erfüllt                               |
| 10011 | TRADE_RETCODE_ERROR              | Fehler der Anforderung Verarbeitung                            |
| 10012 | TRADE_RETCODE_TIMEOUT            | Anforderung abgelehnt nach Zeitablaufs                         |
| 10013 | TRADE_RETCODE_INVALID            | Ungültige Anforderung                                          |
| 10014 | TRADE_RETCODE_INVALID_VOLUME     | Ungültiges Volumen in der Anforderung                          |
| 10015 | TRADE_RETCODE_INVALID_PRICE      | Ungültiger Preis in der Anforderung                            |
| 10016 | TRADE_RETCODE_INVALID_STOPS      | Ungültige Stops in der Anforderung                             |
| 10017 | TRADE_RETCODE_TRADE_DISABLED     | Handel ist verboten                                            |
| 10018 | TRADE_RETCODE_MARKET_CLOSED      | Markt ist geschlossen                                          |
| 10019 | TRADE_RETCODE_NO_MONEY           | Nicht genug Geld, um Anforderung zu erfüllen                   |
| 10020 | TRADE_RETCODE_PRICE_CHANGED      | Preise haben sich verändert                                    |
| 10021 | TRADE_RETCODE_PRICE_OFF          | Es gibt keine Quotationen für die Verarbeitung der Anforderung |
| 10022 | TRADE_RETCODE_INVALID_EXPIRATION | Ungültiges Datum des Orderablaufs in der Anforderung           |

|       |                                  |                                                                                                |
|-------|----------------------------------|------------------------------------------------------------------------------------------------|
| 10023 | TRADE_RETCODE_ORDER_CHANGED      | Order Status hat sich verändert                                                                |
| 10024 | TRADE_RETCODE_TOO_MANY_REQUESTS  | Sehr offene Anforderungen                                                                      |
| 10025 | TRADE_RETCODE_NO_CHANGES         | Keine Veränderungen in der Anforderung                                                         |
| 10026 | TRADE_RETCODE_SERVER_DISABLED_AT | Autotrading ist vom Server untersagt                                                           |
| 10027 | TRADE_RETCODE_CLIENT_DISABLED_AT | Autotrading ist vom Kundenterminal untersagt                                                   |
| 10028 | TRADE_RETCODE_LOCKED             | Anforderung ist für Verarbeitung blockiert                                                     |
| 10029 | TRADE_RETCODE_FROZEN             | Order oder Position sind eingefroren                                                           |
| 10030 | TRADE_RETCODE_INVALID_FILL       | Nicht unterstützter Durchführungstyp der Order nach dem Rest                                   |
| 10031 | TRADE_RETCODE_CONNECTION         | keine Verbindung mit dem Handelsserver                                                         |
| 10032 | TRADE_RETCODE_ONLY_REAL          | Operation ist nur für reelle Kontos erlaubt                                                    |
| 10033 | TRADE_RETCODE_LIMIT_ORDERS       | Limit für Anzahl der Warteordern ist erreicht                                                  |
| 10034 | TRADE_RETCODE_LIMIT_VOLUME       | Limit für Volumen der Ordern und Positionen für dieses Symbol ist erreicht                     |
| 10035 | TRADE_RETCODE_INVALID_ORDER      | Ungültiger oder verbotener <a href="#">Ordertyp</a>                                            |
| 10036 | TRADE_RETCODE_POSITION_CLOSED    | Die Position mit dem angegebenen <a href="#">POSITION_IDENTIFIER</a> wurde bereits geschlossen |

## Compiler Warnungen

Compiler Warnungen haben Informationszwecke und sind keine Fehlermeldungen.

| Nummer | Beschreibung                                                                                                                      |
|--------|-----------------------------------------------------------------------------------------------------------------------------------|
| 21     | Unvollständige Datumaufzeichnung in Zeile datetime                                                                                |
| 22     | Fehlerzahlen in der Zeile datetime für Datum, Anforderungen:<br>Jahr 1970<=X<=3000<br>Monat 0<X<=12<br>Tag 0<X<= 31/30/28(29).... |
| 23     | Fehlerzahlen in der Zeile datetime für Zeit, Anforderungen:<br>Stunde 0<=X<24<br>Minute 0<=X<60                                   |
| 24     | Ungültige Farbe in RGB Format : eine der Komponenten RGB ist weniger als 0 oder mehr als 255                                      |
| 25     | Unbekanntes Symbol der escape Folge.<br>Bekannte: \n \r \t \\ \" \' \X \x                                                         |
| 26     | Volumen der lokalen Variablen der Funktion ist zu gross (>512Kb), reduzieren Sie die Zahl                                         |
| 29     | Aufzählung ist schon definiert (Duplizieren) - Glieder werden zur ersten Definition hinzugefügt                                   |
| 30     | Neudefinieren des Makros                                                                                                          |
| 31     | Variable ist erklärt, aber wird nicht verwendet                                                                                   |
| 32     | Konstruktor muss den Typ void haben                                                                                               |
| 33     | Destruktor muss den Typ void haben                                                                                                |
| 34     | Konstante passt nicht in Bereich von Ganzzahlen (X>_UI64_MAX    X<_I64_MIN) und wird in Typ double umgewandelt                    |
| 35     | Zu langes HEX - mehr als 16 Symbole (ältere Nibbles werden weggeschnitten)                                                        |
| 36     | Es gibt kein einziges Nibble in der HEX Zeile "0x"                                                                                |
| 37     | Es gibt keine einzige Funktion - nichts zu erfüllen                                                                               |
| 38     | verwendet wird eine nicht initialisierte Variable                                                                                 |
| 41     | Funktion hat keinen Körper und wird nicht                                                                                         |

|    |                                                                                                                                         |
|----|-----------------------------------------------------------------------------------------------------------------------------------------|
|    | aufgerufen                                                                                                                              |
| 43 | Mögliche Datenverluste bei Typenreduzierung. Beispiel: <code>int x=(double)z;</code>                                                    |
| 44 | Verlust der Genauigkeit (der Daten) bei der Umwandlung von Konstante. Konstante: <code>int x=M_PI</code>                                |
| 45 | Unterschiede der Zeichen von Operanden in Vergleichsoperationen. Beispiel: <code>(char) c1&gt;(uchar)c2</code>                          |
| 46 | Probleme mit Funktionsimportieren - Erklärung von <code>#import</code> ist angefordert oder Import der Funktionen ist schon geschlossen |
| 47 | Beschreibung ist zu gross - Zusatzsymbole werden nicht in Ausführungsdatei einbeschlossen werden                                        |
| 48 | Anzahl der erklärten Indikatorpuffer ist mehr als angefordert                                                                           |
| 49 | Keine Farbe für Darstellung der graphischen Serie im Indikator                                                                          |
| 50 | Es gibt keine einzige graphische Serie für Indikatordarstellung                                                                         |
| 51 | Funktion-Bearbeiter 'OnStart' ist in Script nicht gefunden                                                                              |
| 52 | Funktion-Bearbeiter 'OnStart' ist mit falschen Parametern definiert                                                                     |
| 53 | Funktion 'OnStart' kann nur in Script definiert werden                                                                                  |
| 54 | Funktion 'OnInit' wird mit falschen Parametern definiert                                                                                |
| 55 | Funktion 'OnInit' wird nicht in Scripts verwendet                                                                                       |
| 56 | Funktion 'OnDeinit' ist mit falschen Parametern definiert                                                                               |
| 57 | Funktion 'OnDeinit' wird nicht in Scripts verwendet                                                                                     |
| 58 | Zwei Funktionen 'OnCalculate' sind definiert. <u><a href="#">OnCalculate()</a> wird auf einem Preisfeld verwendet.</u>                  |
| 59 | Überlauf bei Berechnung der komplizierten <u><a href="#">ganzzahligen</a></u> Konstante ist festgestellt                                |
| 60 | Variable kann <u><a href="#">nicht initialisiert</a></u> sein.                                                                          |

|    |                                                                                                                                                                                                                                      |
|----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 61 | Diese Erklärung macht Aufruf der <a href="#">lokalen Variable</a> , die in der angegebenen Zeile erklärt wird, nicht zugänglich                                                                                                      |
| 62 | Diese Erklärung macht Aufruf der <a href="#">globalen Variable</a> , die in der angegebenen Zeile erklärt wird, nicht zugänglich                                                                                                     |
| 63 | Kann nicht für <a href="#">statische Felder</a> verwendet werden                                                                                                                                                                     |
| 64 | Diese Erklärung macht Aufruf der <a href="#">vordefinierte Variable</a> nicht zugänglich                                                                                                                                             |
| 65 | Der Wert des Ausdrucks ist immer <a href="#">true/false</a>                                                                                                                                                                          |
| 66 | Verwendung einer Variable oder Ausdruck des Typs <a href="#">bool</a> in mathematischen Operationen ist unsicher                                                                                                                     |
| 67 | Das Ergebnis der Anwendung des unären Minus-Operators zum unsigned Typ <a href="#">ulong</a> ist unbestimmt                                                                                                                          |
| 68 | Die Version in der Eigenschaft <a href="#">#property version</a> kann nicht in Sektion <a href="#">Markt</a> verwendet werden, das richtige Format ist <a href="#">#property version "XXX.YYY"</a>                                   |
| 69 | Leere kontrollierte Anweisung aufgetreten                                                                                                                                                                                            |
| 70 | Ungültig Rückgabetyt der Funktion oder falsche Parameter bei der Erklärung einer <a href="#">Event-Handler-Funktion</a>                                                                                                              |
| 71 | Eine explizite <a href="#">Umwandlung von Strukturen</a> zu einem Typ ist erforderlich                                                                                                                                               |
| 72 | Diese Erklärung macht einen direkten Zugang zum Mitglied einer in der angegebenen Zeichenfolge erklärten <a href="#">Klasse</a> unmöglich. Der Zugang wird nur möglich sein, durch <a href="#">Operation mit Kontexterlaubnis ::</a> |
| 73 | Die Konstante in dem binären Schreibung ist zu groß, die höherwertigen Bits werden verworfen werden                                                                                                                                  |
| 74 | Parameter in der Methode der <a href="#">geerbten Klasse</a> variiert sich in der Modifizier <a href="#">const</a> , hat die Kindfunktion die Basisfunktion <a href="#">überladen</a>                                                |
| 75 | Negativer oder zu großer für Offsetwert in <a href="#">Bit-Operation der Verschiebung</a> , ist das Ergebnis undefiniert                                                                                                             |
| 76 | Die Funktion muss einen <a href="#">Wert zurückgeben</a>                                                                                                                                                                             |

|    |                                                                                                                             |
|----|-----------------------------------------------------------------------------------------------------------------------------|
| 77 | Eine Funktion vom Typ <a href="#">void</a> muss keinen Wert zurückgeben                                                     |
| 78 | Nicht alle Ausführungsvarianten geben den Rückgabewert zurück                                                               |
| 79 | Ausdrücke auf der <a href="#">globalen Ebene</a> sind nicht erlaubt                                                         |
| 80 | Mögliche Fehler in der <a href="#">Folge von Operationen</a> , verwenden Sie Klammern um die Reihenfolge explizit anzugeben |
| 81 | Zwei Aufrufe der Funktion <a href="#">OnCalculate()</a> sind gefunden. Die Variante mit Zeitrahmen OHLC wird verwendet      |
| 82 | <a href="#">Die Struktur</a> enthält keine Mitglieder, wird die Größe auf 1 Byte gesetzt                                    |
| 83 | Keine Bearbeitung des Ergebnisses der Funktion                                                                              |

## Kompilierungsfehler

MetaEditor 5, Editor der mql5-Programme, zeigt Fehlermeldungen des Programms, die vom eingebauten Compiler während der Compilierung detektiert werden. Die Liste der Fehler ist in der Tabelle unten angegeben. Für Kompilierung des Ausgangskodes in den ablaufenden Code drücken Sie F7. Programme mit Fehlern können nicht kompiliert werden, bis die vom Compiler angegebene Fehler eliminiert werden.

| Nummer | Beschreibung                                                                        |
|--------|-------------------------------------------------------------------------------------|
| 100    | Dateilesen Fehler                                                                   |
| 101    | Fehler der *.EX5 Dateieröffnung, um sie für Speicherung zu schreiben                |
| 103    | Freier Speicher nicht genug, um Kompilierung zu beenden                             |
| 104    | Leere syntaktische Einheit, die vom Compiler nicht erkannt wird                     |
| 105    | Unkorrekter Dateiname in #include                                                   |
| 106    | Dateizugang in #include (Datei kann schon nicht existieren)                         |
| 108    | Ungeeigneter Name für #define                                                       |
| 109    | Unbekannter Befehl des Preprozessors (gültig #include, #define, #property, #import) |
| 110    | Unbekanntes für Compiler Symbol                                                     |
| 111    | Funktion nicht realisiert (Beschreibung ist vorhanden, Körper gibt es nicht)        |
| 112    | Doppeltes Anführungszeichen ist ausgelassen (")                                     |
| 113    | Öffnende Eckklammer (<) oder doppeltes Anführungszeichen (") ist ausgelassen        |
| 114    | Einzelnes Anführungszeichen ist ausgelassen (')                                     |
| 115    | Schließende Winkelklammer ">" ist ausgelassen                                       |
| 116    | In der Erklärung ist der Typ nicht angegeben                                        |
| 117    | Keine Rückgabeeinweisung return oder nicht in allen Branches der Erfüllung          |
| 118    | Öffnende Klammer der Aufrufparameter wurde erwartet                                 |
| 119    | Schreibfehler EX5                                                                   |
| 120    | Unkorrekter Zugang zum Feldelement                                                  |

|     |                                                                                   |
|-----|-----------------------------------------------------------------------------------|
| 121 | Funktion ist nicht des Typs void und Anweisung return muss den Wert zurückgeben   |
| 122 | Unkorrekte Erklärung des Destruktors                                              |
| 123 | Doppelpunkt fehlt ":"                                                             |
| 124 | Variable ist schon erklärt                                                        |
| 125 | Variable mit diesem Identifikator ist schon erklärt                               |
| 126 | Name der Variable ist zu lang (>250 Symbole)                                      |
| 127 | Struktur mit demselben Identifikator ist schon definiert                          |
| 128 | Struktur ist nicht definiert                                                      |
| 129 | Strukturglied mit demselben Namen ist schon definiert                             |
| 130 | Kein solches Strukturglied                                                        |
| 131 | Paarigkeit der eckigen Klammern ist verletzt                                      |
| 132 | Öffnende runde Klammer "(" ist zu erwarten                                        |
| 133 | Nicht balancierte geschweifte Klammern ( es fehlt ")" )                           |
| 134 | Kompliziert für Kompilierung (zu viele Branchen, interner Levelsteck ist übevoll) |
| 135 | Fehler der Dateieröffnung für Lesen                                               |
| 136 | Nicht genug Speicherplatzes , um Ausgangsdatei in den Speicher zu laden           |
| 137 | Variable ist zu erwarten                                                          |
| 138 | Referenz kann nicht initialisiert werden                                          |
| 140 | Zuordnung war zu erwarten (entsteht bei der Erklärung)                            |
| 141 | Öffnende geschweifte Klammer "{" ist zu erwarten                                  |
| 142 | Parameter kann nur ein <a href="#">dynamisches Feld</a> sein                      |
| 143 | Verwenden des Typs "void" ist ungültig                                            |
| 144 | Kein Paar für ")" oder "]", d.h. es fehlt "(" oder "["                            |
| 145 | Kein Paar für "(" oder "[", d.h. es fehlt ")" oder "]"                            |
| 146 | Unkorrekte Feldgröße                                                              |



|     |                                                                                                     |
|-----|-----------------------------------------------------------------------------------------------------|
| 147 | Zu viele Parameter (>64)                                                                            |
| 149 | Dieses Token ist nicht hier zu erwarten                                                             |
| 150 | Ungültiges Verwenden der Operation (falsche Operanden)                                              |
| 151 | Ausdruck des Typs void ist unzulässig                                                               |
| 152 | Anweisung ist zu erwarten                                                                           |
| 153 | Falsches Verwenden von break                                                                        |
| 154 | Semikolon ist zu erwarten ";"                                                                       |
| 155 | Komma ist zu erwarten ","                                                                           |
| 156 | Typ muss als eine Klasse, nicht als eine Struktur bestimmt werden                                   |
| 157 | Ausdruck ist zu erwarten                                                                            |
| 158 | In HEX tritt "nicht HEX Symbol" auf oder zu lange Zahl (Zifferanzahl > 511)                         |
| 159 | Zeile-Konstante hat mehr als 65534 Symbole                                                          |
| 160 | Funktionsdefinieren ist hier unzugänglich                                                           |
| 161 | Unerwartetes Programmende                                                                           |
| 162 | Vorrangige Erklärung für Strukturen ist verboten                                                    |
| 163 | Funktion mit demselben Namen ist schon definiert und hat einen anderen Typ des rückgegebenen Wertes |
| 164 | Funktion mit demselben Namen ist schon definiert und hat einen anderen Parametervorrat              |
| 165 | Funktion mit demselben Namen ist schon definiert und implementiert                                  |
| 166 | Überladen der Funktion für diesen Aufruf ist nicht gefunden                                         |
| 167 | Funktion mit dem Rückgabewert des Typs void kann den Wert nicht rückgeben                           |
| 168 | Funktion ist nicht definiert                                                                        |
| 170 | Wert ist zu erwarten                                                                                |
| 171 | Im Ausdruck case sind nur ganzzahlige Konstanten gültig                                             |
| 172 | Wert für case in diesem switch wurde schon verwendet                                                |

|     |                                                                                |
|-----|--------------------------------------------------------------------------------|
| 173 | Ganzzahliger Wert ist zu erwarten                                              |
| 174 | Im Ausdruck #import ist Dateiname zu erwarten                                  |
| 175 | Ausdrücke sind nicht auf dem globalen Niveau zugelassen                        |
| 176 | Runde Klammer ")" vor ";" ist ausgelassen                                      |
| 177 | Links von Gleichheitszeichen ist die Variable erwartet                         |
| 178 | Ausdrucksergebnis wird nicht verwendet                                         |
| 179 | Variablenerklärung in case ist nicht zugelassen                                |
| 180 | Implizite Umwandlung der Zeile in die Zahl                                     |
| 181 | Implizite Umwandlung der Zahl in die Zeile                                     |
| 182 | Nichteindeutiger Aufruf der überladenen Funktion (mehrere Überladungen passen) |
| 183 | Ungültiges else ohne entsprechendes if                                         |
| 184 | Ungültiges case oder default ohne entsprechendes switch                        |
| 185 | Ungültiges Verwenden der Ellipse                                               |
| 186 | Initialisierungsfolge hat mehr Elementen als Initialisierungsvariable          |
| 187 | Konstante für case ist zu erwarten                                             |
| 188 | Konstantausdruck ist angefordert                                               |
| 189 | Konstantvariable kann nicht verändert werden                                   |
| 190 | Rechte Klammer oder Komma ist zu erwarten (Erklärung des Feldgliedes)          |
| 191 | Enum ID wird bereits verwendet                                                 |
| 192 | Aufzählung kann keine Zugangsmodifikatoren haben (const, extern, static)       |
| 193 | Aufzählungsglied ist schon mit dem anderen Wert erklärt                        |
| 194 | Es gibt eine Variable, die mit demselben Namen definiert ist                   |
| 195 | Es gibt eine Struktur, die mit demselben Namen definiert wird                  |
| 196 | Name des Aufzählungsgliedes ist zu erwarten                                    |
| 197 | Ganzzahliger Ausdruck ist zu erwarten                                          |

|     |                                                                                                      |
|-----|------------------------------------------------------------------------------------------------------|
| 198 | Dividieren durch Null im Konstantausdruck                                                            |
| 199 | Falsche Parameterzahl in der Funktion                                                                |
| 200 | Parameter per Referenz muss Variable sein                                                            |
| 201 | Variable desselben Typs für Übertragung durch Referenz wird erwartet                                 |
| 202 | Konstantvariable kann nicht durch nicht Konstantreferenz übertragen werden                           |
| 203 | Ganzzahlige positive Konstante ist angefordert                                                       |
| 204 | Fehler des Zugangs zum geschützten Glied der Klasse                                                  |
| 205 | Import ist schon anders definiert                                                                    |
| 208 | Durchführungsdatei nicht erzeugt                                                                     |
| 209 | für Indikator ist kein Eingangspunkt gefunden 'OnCalculate'                                          |
| 210 | Anweisung continue kann nur innerhalb des Zyklus verwendet werden                                    |
| 211 | Fehler des Zuganges zum private(geschlossenen) Glied der Klasse                                      |
| 213 | Strukturmethode oder Klassenmethode ist nicht erklärt                                                |
| 214 | Fehler des Zugangs zur private(geschlossenen) Methode der Klasse                                     |
| 216 | Kopieren der Strukturen mit Objekten ist nicht zulässig                                              |
| 218 | Index außerhalb der Feldgrenzen                                                                      |
| 219 | Felderinitialisierung in Erklärung der Struktur oder der Klasse ist ungültig                         |
| 220 | Konstruktor der Klasse kann keine Parameter haben                                                    |
| 221 | Destruktor der Klasse kann keine Parameter haben                                                     |
| 222 | Methode der Klasse oder der Struktur mit denselben Parametern ist schon erklärt                      |
| 223 | Operand ist zu erwarten                                                                              |
| 224 | Es gibt schon die Methode mit diesem Namen, aber mit anderen Parametern (Erklärung!=Implementierung) |
| 225 | Zu importierende Funktion ist nicht                                                                  |

|     |                                                                                                          |
|-----|----------------------------------------------------------------------------------------------------------|
|     | beschrieben                                                                                              |
| 226 | Функция <a href="#">ZeroMemory()</a> не применима для классов с защищенными членами или наследованием    |
| 227 | Unklarer Aufruf der überladenen Funktion (exaktes Zusammenfallen der Parameter für mehrere Überladungen) |
| 228 | Variablenname ist zu erwarten                                                                            |
| 229 | Referenz kann nicht an dieser Stelle erklärt werden                                                      |
| 230 | Schon verwendet als Aufzählungsname                                                                      |
| 232 | Klasse oder Struktur ist zu erwarten                                                                     |
| 235 | delete kann nicht für Feldentfernung aufgerufen werden                                                   |
| 236 | Anweisung 'while' ist zu erwarten                                                                        |
| 237 | In delete muss den Anzeiger geben                                                                        |
| 238 | Es gibt schon default für dieses switch                                                                  |
| 239 | Syntaktischer Fehler                                                                                     |
| 240 | Escape-Folge kann nur in Zeilen auftreten (fängt mit '\ ' an)                                            |
| 241 | Feld angefordert - Eckige Klammer '[' gehört zum Feld nicht oder oder keine Felder als Feldparameters    |
| 242 | Kann nicht durch initialisierende Folge initialisiert werden                                             |
| 243 | Import ist nicht definiert                                                                               |
| 244 | Fehler des Optimierers auf dem syntaktischen Baum                                                        |
| 245 | Zu viele Strukturen sind erklärt (versuchen Sie das Programm zu vereinfachen)                            |
| 246 | Parameterumwandlung ist ungültig                                                                         |
| 247 | Unkorrektes Verwenden der Anweisung delete                                                               |
| 248 | Anzeiger für Referenz kann nicht erklärt werden                                                          |
| 249 | Es ist nicht zugelassen Referenz für Referenz zu erklären                                                |
| 250 | Es ist nicht zugelassen, Anzeiger für Anzeiger zu erklären                                               |

|     |                                                                                                          |
|-----|----------------------------------------------------------------------------------------------------------|
| 251 | Strukturerklärung in der Parameterliste ist ungültig                                                     |
| 252 | Ungültige Operation der Typenreduzierung                                                                 |
| 253 | Anzeiger kann nur für Klasse oder Struktur erklärt werden                                                |
| 256 | Nicht erklärter Identifikator                                                                            |
| 257 | Fehler des Optimierers des Durchführungskodes                                                            |
| 258 | Fehler der Generierung des Durchführungskodes                                                            |
| 260 | Ungültiger Ausdruck für Anweisung switch                                                                 |
| 261 | Pool der Zeilenkonstanten ist überfüllt, vereinfachen Sie das Programm                                   |
| 262 | Unmöglich zur Aufzählung zu reduzieren                                                                   |
| 263 | <i>virtual</i> kann nicht für Daten verwendet werden (Glieder der Klasse oder der Struktur)              |
| 264 | Geschützte Methode der Klasse kann nicht aufgerufen werden                                               |
| 265 | Undefinierte virtuelle Funktion gibt anderen Typ zurück                                                  |
| 266 | Klasse kann nicht von der Struktur geerbt werden                                                         |
| 267 | Struktur kann nicht von der Klasse geerbt werden                                                         |
| 268 | Konstruktor kann nicht virtuell sein (Spezifikator <i>virtual</i> ist ungültig)                          |
| 269 | Struktur kann keine virtuelle Methoden haben                                                             |
| 270 | Funktion muss den Körper haben                                                                           |
| 271 | Überladen der Systemfunktionen (Funktionen des Terminals) ist verboten                                   |
| 272 | Spezifikator <i>const</i> ist für die Funktionen, die keine Klassen- oder Strukturglieder sind, ungültig |
| 274 | Klassenglieder können nicht in der Konstantmethode geändert werden                                       |
| 276 | Unpassende Initialisierungsfolge                                                                         |
| 277 | Defaultwert für Parameter ist ausgelassen (Spezifik der Erklärung von Defaultparametern)                 |

|     |                                                                                                                                                                                     |
|-----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 278 | Neubestimmen des Default-Parameters (in der Erklärung und Realisierung sind die Werte verschieden)                                                                                  |
| 279 | Nicht-Konstantmethode für Konstantobjekt kann nicht aufgerufen werden                                                                                                               |
| 280 | Für Zugang zu Gliedern wird das Objekt gefordert (Punkt für eine Nichtklasse/Struktur gestellt)                                                                                     |
| 281 | Name der schon erklärten Struktur kann nicht bei der Erklärung verwendet werden                                                                                                     |
| 284 | Nicht autorisierte Umwandlung (bei der geschlossenen Vererbung)                                                                                                                     |
| 285 | Strukturen und Felder können nicht als input-Variablen verwendet werden                                                                                                             |
| 286 | Spezifikator const ist für Konstruktor/ Destruktor ungültig                                                                                                                         |
| 287 | Falsche Zeilenausdruck für Typ datetime                                                                                                                                             |
| 288 | unbekannte Eigenschaft (#property)                                                                                                                                                  |
| 289 | unkorrektter Wert für Eigenschaft                                                                                                                                                   |
| 290 | unkorrektter Index für Eigenschaft in #property                                                                                                                                     |
| 291 | Anrufparameter ist ausgelassen - < func(x,) >                                                                                                                                       |
| 293 | Objekt muss per Referenz übertragen werden                                                                                                                                          |
| 294 | Feld muss per Referenz übertragen werden                                                                                                                                            |
| 295 | Funktion wurde als zu exportierende Funktion erklärt                                                                                                                                |
| 296 | Funktion wurde nicht als zu exportierende Funktion erklärt                                                                                                                          |
| 297 | Zu importierende Funktion kann nicht exportiert werden                                                                                                                              |
| 298 | Zu importierende Funktion kann nicht diesen Parameter haben (Anzeiger, Klasse oder Struktur, die dynamisches Feld, Anzeiger, Klasse usw. enthalten, können nicht übertragen werden) |
| 299 | Muss eine Klassen sein                                                                                                                                                              |
| 300 | #import ist nicht abgeschlossen                                                                                                                                                     |
| 302 | Nicht-Kompatibilität der Typen                                                                                                                                                      |
| 303 | <a href="#">extern</a> -Variable ist schon initialisiert                                                                                                                            |

|     |                                                                                                            |
|-----|------------------------------------------------------------------------------------------------------------|
| 304 | Keine einzige <a href="#">exportierte</a> Funktion oder <a href="#">Standardeingangspunkt</a> ist gefunden |
| 305 | Expliziter Aufruf des <a href="#">Konstruktors</a> ist verboten                                            |
| 306 | Methode wurde als <a href="#">konstant</a> erklärt                                                         |
| 307 | Methode wurde nicht als <a href="#">konstant</a> erklärt                                                   |
| 308 | Unkorrekte Größe der Ressource-Datei                                                                       |
| 309 | Unkorrekter Ressource-Name                                                                                 |
| 310 | Fehler beim Öffnen der Ressource-Datei                                                                     |
| 311 | Fehler beim Lesen der Ressource-Datei                                                                      |
| 312 | Unbekannter Typ der Ressource                                                                              |
| 313 | Unkorrekte Pfad zur Ressource-Datei                                                                        |
| 314 | Der angegebene <a href="#">Ressource</a> -Name wird bereits verwendet                                      |
| 315 | Erwartete Parameter des Makros                                                                             |
| 316 | Nach dem Makronamen muss ein Leerzeichen stehen                                                            |
| 317 | Fehler in der formalen Parameter des Makros                                                                |
| 318 | Falsche Anzahl von Parametern bei der Verwendung eines Makros                                              |
| 319 | Zu viele Parameter für ein Makro                                                                           |
| 320 | Zu komplex, das Makro soll vereinfacht werden                                                              |
| 321 | Parameter für <a href="#">EnumToString()</a> kann nur Aufzählung sein                                      |
| 322 | <a href="#">Ressourcenname</a> ist zu lang                                                                 |
| 323 | Bildformate ist nicht unterstützt (nur BMP mit 24 oder 32 Bit Farbtiefe ist erlaubt)                       |
| 324 | Ein Array kann nicht im Operator erklärt werden                                                            |
| 325 | Die Funktion kann nur auf <a href="#">globalen</a> Bereich erklärt werden                                  |
| 326 | Die Erklärung ist nicht für den aktuellen <a href="#">Sichtbereich</a> erlaubt                             |
| 327 | Initialisierung der statischen Variablen mit Werte der lokalen Variablen ist ungültig                      |
| 328 | Unerlaubte Deklaration der Array von Objekten, die keinen <a href="#">Standardkonstruktor</a> haben        |
| 329 | <a href="#">Initialisierungsliste</a> ist nur für <a href="#">Konstruktoren</a> erlaubt                    |

|     |                                                                                                                                           |
|-----|-------------------------------------------------------------------------------------------------------------------------------------------|
| 330 | Es gibt keine Definition der Function nach der <a href="#">Initialisierungsliste</a>                                                      |
| 331 | <a href="#">Die Initialisierungsliste</a> ist leer                                                                                        |
| 332 | Array-Initialisierung im Konstruktor ist verboten                                                                                         |
| 333 | Initialisierung der Mitglieder der Elternklasse in der <a href="#">Initialisierungsliste</a> ist verboten                                 |
| 334 | Ein Ausdruck des <a href="#">Typs Integer</a> wurde erwartet                                                                              |
| 335 | Erforderlicher Speicher für das <a href="#">Array</a> übersteigt den maximalen Wert                                                       |
| 336 | Erforderlicher Speicher für die <a href="#">Struktur</a> übersteigt den maximalen Wert                                                    |
| 337 | Erforderlicher Speicher für die auf <a href="#">globaler Ebene</a> deklarierte Variablen übersteigt den maximalen Wert                    |
| 338 | Erforderlicher Speicher für die <a href="#">lokale Variablen</a> übersteigt den maximalen Wert                                            |
| 339 | <a href="#">Der Konstruktor</a> ist nicht definiert                                                                                       |
| 340 | Ungültige Name der <a href="#">Symboldatei</a>                                                                                            |
| 341 | Konnte nicht die <a href="#">Symboldatei</a> unter dem angegebenen Pfad öffnen                                                            |
| 342 | Ungültig <a href="#">Symboldateiformat</a> , es muss <a href="#">ICO</a> sein                                                             |
| 343 | Neuinitialisierung eines Mitglieds in einer Klasse/Struktur Konstruktor mit der <a href="#">Initialisierungsliste</a>                     |
| 344 | Initialisierung der <a href="#">statischen</a> Mitglieder in der <a href="#">Initialisierungsliste</a> des Konstruktors ist nicht erlaubt |
| 345 | Initialisierung der <a href="#">nichtstatischen</a> Mitglied einer Klasse/Struktur auf <a href="#">globaler Ebene</a> ist nicht erlaubt   |
| 346 | Der Name der Methode der Klasse/Struktur stimmt mit der zuvor deklarierten Namen eines Mitglieds                                          |
| 347 | Der Name des Mitglieds der Klasse/Struktur stimmt mit der zuvor deklarierten Namen einer Methode                                          |
| 348 | <a href="#">Virtuelle</a> Funktion kann nicht als <a href="#">static</a> deklariert werden                                                |
| 349 | Modifikator <a href="#">const</a> ist für eine <a href="#">statische</a> Funktion nicht erlaubt                                           |



|     |                                                                                                                                                |
|-----|------------------------------------------------------------------------------------------------------------------------------------------------|
| 350 | <a href="#">Konstruktor</a> oder <a href="#">Destruktor</a> können nicht statisch sein                                                         |
| 351 | Nicht-statischer Mitglied/Methode einer Klasse oder einer Struktur kann nicht von einer <a href="#">statischen</a> Funktion zugegriffen werden |
| 352 | Nach dem Schlüsselwort <a href="#">operator</a> wurde eine überladene Operation (+, -, [], ++, -- usw.) erwartet                               |
| 353 | Nicht alle Operationen können in MQL5 <a href="#">überladen</a> werden                                                                         |
| 354 | Definition entspricht nicht der Deklaration                                                                                                    |
| 355 | Eine ungültige Anzahl der Parameter ist für den <a href="#">Operator</a> angegeben                                                             |
| 356 | <a href="#">Event-Handling-Funktion</a> nicht gefunden                                                                                         |
| 357 | Methoden können nicht <a href="#">exportiert sein</a>                                                                                          |
| 358 | Ein Zeiger auf das <a href="#">konstante</a> Objekt kann nicht in einen Zeiger auf ein nicht-konstantes Objekt umwandelt werden                |
| 359 | Klasse-Schablonen werden noch nicht unterstützt                                                                                                |
| 360 | <a href="#">Überladen</a> von Funktionsschablonen wird noch nicht unterstützt                                                                  |
| 361 | Sie können nicht eine Funktionsschablone benutzen                                                                                              |
| 362 | Unklare Parameter in einer Funktionsschablone (verschiedene Typen von Parametern passen)                                                       |
| 363 | Es ist unmöglich zu bestimmen, in welchen Parametertyp den Argument von Funktionsschablone umzuwandeln                                         |
| 364 | Falsche Anzahl von Parametern in einer Funktionsschablone                                                                                      |
| 365 | Funktionsschablone kann nicht <a href="#">virtuell</a> sein                                                                                    |
| 366 | Funktionsschablonen können nicht exportiert werden                                                                                             |
| 367 | Funktionsschablonen können nicht importiert werden                                                                                             |
| 368 | Strukturen mit Objekten nicht akzeptabel sind                                                                                                  |
| 369 | String Arrays und Strukturen, die Objekte enthalten, sind ungültig                                                                             |

|     |                                                                                                         |
|-----|---------------------------------------------------------------------------------------------------------|
| 370 | <u>Ein Statisches Mitglied der Klasse/Struktur</u> soll explizit initialisiert werden                   |
| 371 | Die Beschränkung der Compiler: eine Ziele kann nicht mehr als 65.535 Zeichen enthalten                  |
| 372 | Inkonsistent <u>#ifdef/#endif</u>                                                                       |
| 373 | Результатом выполнения функции не может быть объект класса, так как отсутствует конструктор копирования |
| 374 | Нельзя использовать нестатические члены и/или методы при инициализации статической переменной           |
| 375 | OnTesterInit() нельзя использовать без объявления обработчика OnTesterDeinit()                          |
| 376 | Имя локальной переменной совпадает с именем одного из параметров функции                                |
| 377 | Нельзя использовать макросы <u>__FUNCSIG__</u> и <u>__FUNCTION__</u> вне тела функции                   |

## Ausführungsfehler

GetLastError() - Funktion, die den Code des letzten Fehlers rückgibt, die in der vorbestimmten Variable \_LastError aufbewahren wird. Der Wert dieser Variable muss durch die Funktion ResetLastError() auf Null gestellt werden.

| Konstante                      | Wert | Beschreibung                                                                                                                 |
|--------------------------------|------|------------------------------------------------------------------------------------------------------------------------------|
| ERR_INTERNAL_ERROR             | 4001 | Unerwarteter interner Fehler                                                                                                 |
| ERR_WRONG_INTERNAL_PARAMETER   | 4002 | Fehlerparameter beim internen Aufruf des Kundenterminals                                                                     |
| ERR_INVALID_PARAMETER          | 4003 | Fehlerparameter beim Aufruf der Systemfunktion                                                                               |
| ERR_NOT_ENOUGH_MEMORY          | 4004 | Nicht genug Speicherplatz für Ausführung der Systemfunktion                                                                  |
| ERR_STRUCT_WITHOBJECTS_ORCLASS | 4005 | Struktur hat Zeilenobjekte und/oder Objekte der dynamischen Felder und/oder Strukturen mit solchen Objekten und/oder Klassen |
| ERR_INVALID_ARRAY              | 4006 | Feld des ungültigen Typs, der ungültiger Größe oder beschädigtes Objekt des dynamischen Feldes                               |
| ERR_ARRAY_RESIZE_ERROR         | 4007 | nicht genug Speicherplatz für Neuordnung des Feldes oder der Versuch der Größenveränderung des statischen Feldes             |
| ERR_STRING_RESIZE_ERROR        | 4008 | Nicht genug Speicherplatz für Neuordnung der Zeile                                                                           |
| ERR_NOTINITIALIZED_STRING      | 4009 | nicht initialisierte Zeile                                                                                                   |
| ERR_INVALID_DATETIME           | 4010 | Ungültiger Wert des Datums und/oder der Zeit An                                                                              |
| ERR_ARRAY_BAD_SIZE             | 4011 | Angeforderte Feldgröße ist mehr als 2 Gigabyte                                                                               |
| ERR_INVALID_POINTER            | 4012 | Ungültiger Anzeiger                                                                                                          |
| ERR_INVALID_POINTER_TYPE       | 4013 | Falscher Typ des Anzeigers                                                                                                   |
| ERR_FUNCTION_NOT_ALLOWED       | 4014 | Systemfunktion ist für Anruf nicht erlaubt                                                                                   |
| ERR_RESOURCE_NAME_DUPLIC       | 4015 | Die Namen der <u>dynamischen</u>                                                                                             |

|                                |      |                                                                                            |
|--------------------------------|------|--------------------------------------------------------------------------------------------|
| ATED                           |      | und <a href="#">statischen</a> Ressource sind gleich                                       |
| ERR_RESOURCE_NOT_FOUND         | 4016 | Ressource mit diesem Namen ist nicht in EX5 gefunden                                       |
| ERR_RESOURCE_UNSUPPORTED_TYPE  | 4017 | Nicht unterstützte Ressourcetyp oder Größe von mehr als 16 MB                              |
| ERR_RESOURCE_NAME_IS_TOO_LONG  | 4018 | Der Name der Ressource ist mehr als 63 Zeichen                                             |
| <b>Charts</b>                  |      |                                                                                            |
| ERR_CHART_WRONG_ID             | 4101 | Falscher Identifikator des Charts                                                          |
| ERR_CHART_NO_REPLY             | 4102 | Chart antwortet nicht                                                                      |
| ERR_CHART_NOT_FOUND            | 4103 | Chart ist nicht gefunden                                                                   |
| ERR_CHART_NO_EXPERT            | 4104 | Im Chart gibt es keinen Expert Advisor, der das Ereignis verarbeiten kann                  |
| ERR_CHART_CANNOT_OPEN          | 4105 | Fehler bei Charteröffnung                                                                  |
| ERR_CHART_CANNOT_CHANGE        | 4106 | Fehler bei Veränderung des Symbols und der Periode des Charts                              |
| ERR_CHART_WRONG_PARAMETER      | 4107 | Ein fehlerhafter Parameterwert für <a href="#">Funktionen für die Arbeit mit dem Chart</a> |
| ERR_CHART_CANNOT_CREATE_TIMER  | 4108 | Fehler bei Timererzeugung                                                                  |
| ERR_CHART_WRONG_PROPERTY       | 4109 | Falsche Charteigenschaft ID                                                                |
| ERR_CHART_SCREENSHOT_FAILED    | 4110 | Fehler bei Screenshot Erzeugung                                                            |
| ERR_CHART_NAVIGATE_FAILED      | 4111 | Fehler der Navigierung durch Chart                                                         |
| ERR_CHART_TEMPLATE_FAILED      | 4112 | Fehler der Schablonenverwendung                                                            |
| ERR_CHART_WINDOW_NOT_FOUND     | 4113 | Subfenster mit angegebenen Indikator nicht gefunden                                        |
| ERR_CHART_INDICATOR_CANNOT_ADD | 4114 | Fehler beim Hinzufügen eines Indikators zu einem Chart                                     |
| ERR_CHART_INDICATOR_CANNOT_DEL | 4115 | Fehler beim Entfernen des Indikators aus dem Chart                                         |

|                               |      |                                                                    |
|-------------------------------|------|--------------------------------------------------------------------|
| ERR_CHART_INDICATOR_NOT_FOUND | 4116 | Der Indikator kann nicht auf dem angegebenen Chart gefunden werden |
| <b>Graphische Objekte</b>     |      |                                                                    |
| ERR_OBJECT_ERROR              | 4201 | Fehler bei Arbeit mit dem graphischen Objekt                       |
| ERR_OBJECT_NOT_FOUND          | 4202 | Graphisches Objekt ist nicht gefunden                              |
| ERR_OBJECT_WRONG_PROPERTY     | 4203 | Fehleridentifikator der Eigenschaft des graphischen Objektes       |
| ERR_OBJECT_GETDATE_FAILED     | 4204 | Datum entsprechend dem Wert kann nicht erhalten werden             |
| ERR_OBJECT_GETVALUE_FAILED    | 4205 | Wert entsprechend dem Datum kann nicht erhalten werden             |
| <b>MarketInfo</b>             |      |                                                                    |
| ERR_MARKET_UNKNOWN_SYMBOL     | 4301 | Unbekanntes Symbol                                                 |
| ERR_MARKET_NOT_SELECTED       | 4302 | Symbol nicht gewählt in MarketWatch                                |
| ERR_MARKET_WRONG_PROPERTY     | 4303 | Fehleridentifikator der Symboleigenschaft                          |
| ERR_MARKET_LASTTIME_UNKNOWN   | 4304 | Zeit des Letzten Ticks ist unbekannt (es gab keine Ticks)          |
| ERR_MARKET_SELECT_ERROR       | 4305 | Fehler bei Hinzufügung oder Löschung eines Symbols in MarketWatch  |
| <b>Zugang zur Geschichte</b>  |      |                                                                    |
| ERR_HISTORY_NOT_FOUND         | 4401 | Angeforderte Geschichte ist nicht gefunden                         |
| ERR_HISTORY_WRONG_PROPERTY    | 4402 | Falscher Identifikator der Geschichtseigenschaft                   |
| <b>Global_Variables</b>       |      |                                                                    |
| ERR_GLOBALVARIABLE_NOT_FOUND  | 4501 | Globale Variable des Kundenterminals ist nicht gefunden            |
| ERR_GLOBALVARIABLE_EXISTS     | 4502 | Globale Variable des Kundenterminals mit demselben Namen existiert |

|                                              |      |                                                                                                                                                                   |
|----------------------------------------------|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                              |      | schon                                                                                                                                                             |
| ERR_MAIL_SEND_FAILED                         | 4510 | Fehler beim Mailsenden                                                                                                                                            |
| ERR_PLAY_SOUND_FAILED                        | 4511 | Fehler beim Lautabspielen                                                                                                                                         |
| ERR_MQL5_WRONG_PROPERTY                      | 4512 | Falscher Identifikator der Programmeigenschaft                                                                                                                    |
| ERR_TERMINAL_WRONG_PROPERTY                  | 4513 | Falscher Identifikator der Terminaleigenschaft                                                                                                                    |
| ERR_FTP_SEND_FAILED                          | 4514 | Fehler beim Dateisenden via ftp                                                                                                                                   |
| ERR_NOTIFICATION_SEND_FAILED                 | 4515 | <a href="#">Benachrichtigung</a> konnte nicht gesendet werden                                                                                                     |
| ERR_NOTIFICATION_WRONG_PARAMETER             | 4516 | Ungültige Parameter, um Benachrichtigung zu senden - ein leerer String oder <a href="#">NULL</a> war in die Funktion <a href="#">SendNotification()</a> übergeben |
| ERR_NOTIFICATION_WRONG_SETTINGS              | 4517 | Falsche Einstellung von Benachrichtigungen im Terminal (ID ist nicht eingegeben oder keine Berechtigung ist aufgesetzt)                                           |
| ERR_NOTIFICATION_TOO_FREQUENT                | 4518 | Zu häufige Zusendung von Benachrichtigungen                                                                                                                       |
| <b>Puffer der Benutzerindikatoren</b>        |      |                                                                                                                                                                   |
| ERR_BUFFERS_NO_MEMORY                        | 4601 | Nicht genug Speicherplatz für Neuordnung der Indikatorpuffer                                                                                                      |
| ERR_BUFFERS_WRONG_INDEX                      | 4602 | Falscher Index des Indikatorpuffers                                                                                                                               |
| <b>Eigenschaften der Benutzerindikatoren</b> |      |                                                                                                                                                                   |
| ERR_CUSTOM_WRONG_PROPERTY                    | 4603 | Falscher Identifikator der Eigenschaft von Benutzerindikator                                                                                                      |
| <b>Account</b>                               |      |                                                                                                                                                                   |
| ERR_ACCOUNT_WRONG_PROPERTY                   | 4701 | Falscher Identifikator der Kontoeigenschaft                                                                                                                       |
| ERR_TRADE_WRONG_PROPERTY                     | 4751 | Falscher Identifikator der Handelseigenschaft                                                                                                                     |

|                                  |      |                                                                      |
|----------------------------------|------|----------------------------------------------------------------------|
| ERR_TRADE_DISABLED               | 4752 | Handel für Expert ist verboten                                       |
| ERR_TRADE_POSITION_NOT_FOUND     | 4753 | Position ist nicht gefunden                                          |
| ERR_TRADE_ORDER_NOT_FOUND        | 4754 | Order ist nicht gefunden                                             |
| ERR_TRADE_DEAL_NOT_FOUND         | 4755 | Deal ist nicht gefunden                                              |
| ERR_TRADE_SEND_FAILED            | 4756 | Fehler beim Senden der Handelsanforderung                            |
| <b>Indikator</b>                 |      |                                                                      |
| ERR_INDICATOR_UNKNOWN_SYMBOL     | 4801 | Unbekanntes Symbol                                                   |
| ERR_INDICATOR_CANNOT_CREATE      | 4802 | Indikator kann nicht erzeugt werden                                  |
| ERR_INDICATOR_NO_MEMORY          | 4803 | Nicht genug Speicherplatz für Indikatorzufügung                      |
| ERR_INDICATOR_CANNOT_APPLY       | 4804 | Indikator kann nicht zum anderen Indikator angewendet werden         |
| ERR_INDICATOR_CANNOT_ADD         | 4805 | Fehler bei Indikatorzufügung                                         |
| ERR_INDICATOR_DATA_NOT_FOUND     | 4806 | Angeforderte Daten nicht gefunden                                    |
| ERR_INDICATOR_WRONG_HANDLE       | 4807 | Fehlerhandle des Indikators                                          |
| ERR_INDICATOR_WRONG_PARAMETERS   | 4808 | Falsche Parameterzahl bei Anzeiger Erzeugung                         |
| ERR_INDICATOR_PARAMETERS_MISSING | 4809 | Keine Parameter bei Anzeiger Erzeugung                               |
| ERR_INDICATOR_CUSTOM_NAME        | 4810 | Der erste Parameter im Feld muss der Name des Benutzeranzeigers sein |
| ERR_INDICATOR_PARAMETER_TYPE     | 4811 | Falscher Parametertyp im Feld bei Anzeiger Erzeugung                 |
| ERR_INDICATOR_WRONG_INDEX        | 4812 | Fehlerindex des angeforderten Indikatorpuffers                       |
| <b>Depth of Market</b>           |      |                                                                      |
| ERR_BOOKS_CANNOT_ADD             | 4901 | DOM kann nicht zugefügt werden                                       |
| ERR_BOOKS_CANNOT_DELETE          | 4902 | DOM kann nicht entfernt                                              |

|                            |      |                                                                                    |
|----------------------------|------|------------------------------------------------------------------------------------|
|                            |      | werden                                                                             |
| ERR_BOOKS_CANNOT_GET       | 4903 | DOM Dasten können nicht erhalten werden                                            |
| ERR_BOOKS_CANNOT_SUBSCRIBE | 4904 | Fehler in Subskription neue DOM Daten zu erhalten                                  |
| <b>Dateioperationen</b>    |      |                                                                                    |
| ERR_TOO_MANY_FILES         | 5001 | Mehr als 64 Dateien können nicht gleichzeitig eröffnet werden                      |
| ERR_WRONG_FILENAME         | 5002 | Ungültiger Dateiname                                                               |
| ERR_TOO_LONG_FILENAME      | 5003 | Zu langer Dateiname                                                                |
| ERR_CANNOT_OPEN_FILE       | 5004 | Fehler der Dateieröffnung                                                          |
| ERR_FILE_CACHEBUFFER_ERROR | 5005 | Nicht genug Speicherplatz, um Cache zu lesen                                       |
| ERR_CANNOT_DELETE_FILE     | 5006 | Fehler der Dateientfernung                                                         |
| ERR_INVALID_FILEHANDLE     | 5007 | Datei mit diesem handle ist schon geschlossen oder war ganz nicht eröffnet         |
| ERR_WRONG_FILEHANDLE       | 5008 | Falsches handle der Datei                                                          |
| ERR_FILE_NOTTOWRITE        | 5009 | Datei muss für Schreiben eröffnet werden                                           |
| ERR_FILE_NOTTOREAD         | 5010 | Datei muss für Lesen eröffnet werden                                               |
| ERR_FILE_NOTBIN            | 5011 | Datei muss als binäre Datei eröffnet werden                                        |
| ERR_FILE_NOTTXT            | 5012 | Datei muss als Textdatei eröffnet werden                                           |
| ERR_FILE_NOTTXTORCSV       | 5013 | Datei muss als Textdatei oder CSV eröffnet werden                                  |
| ERR_FILE_NOTCSV            | 5014 | Datei muss als CSV eröffnet werden                                                 |
| ERR_FILE_READERROR         | 5015 | Fehler des Datei Lesen                                                             |
| ERR_FILE_BINSTRINGSIZE     | 5016 | Es muss Zeilengröße angegeben werden, denn die Datei ist als binäre Datei eröffnet |
| ERR_INCOMPATIBLE_FILE      | 5017 | für Zeilenfelder muss Textdatei sein, für andere - binäre Datei                    |



|                             |      |                                                                                                                                        |
|-----------------------------|------|----------------------------------------------------------------------------------------------------------------------------------------|
| ERR_FILE_IS_DIRECTORY       | 5018 | Das ist keine Datei, sondern ein Verzeichnis                                                                                           |
| ERR_FILE_NOT_EXIST          | 5019 | Datei existiert nicht                                                                                                                  |
| ERR_FILE_CANNOT_REWRITE     | 5020 | Datei kann nicht neugeschrieben werden                                                                                                 |
| ERR_WRONG_DIRECTORYNAME     | 5021 | Falscher Verzeichnisname                                                                                                               |
| ERR_DIRECTORY_NOT_EXIST     | 5022 | Verzeichnis existiert nicht                                                                                                            |
| ERR_FILE_ISNOT_DIRECTORY    | 5023 | Das ist eine Datei, nicht Verzeichnis                                                                                                  |
| ERR_CANNOT_DELETE_DIRECTORY | 5024 | Verzeichnis kann nicht entfernt werden                                                                                                 |
| ERR_CANNOT_CLEAN_DIRECTORY  | 5025 | Misslungen, das Verzeichnis zu löschen (eine oder mehrere Dateien können blockiert werden und Operation der Entfernung ist misslungen) |
| ERR_FILE_WRITEERROR         | 5026 | Ressource konnte nicht in die Datei geschrieben werden                                                                                 |
| <b>Zeilenreduzierung</b>    |      |                                                                                                                                        |
| ERR_NO_STRING_DATE          | 5030 | In der Zeile gibt es kein Datum                                                                                                        |
| ERR_WRONG_STRING_DATE       | 5031 | Falsches Datum in der Zeile                                                                                                            |
| ERR_WRONG_STRING_TIME       | 5032 | Falsche Zeit in der Zeile                                                                                                              |
| ERR_STRING_TIME_ERROR       | 5033 | Fehler der Umwandlung der Zeile in Datum                                                                                               |
| ERR_STRING_OUT_OF_MEMORY    | 5034 | Nicht genug Speicherplatz für Zeile                                                                                                    |
| ERR_STRING_SMALL_LEN        | 5035 | Zeilenlänge ist kleiner als erwartet                                                                                                   |
| ERR_STRING_TOO_BIGNUMBER    | 5036 | Zu lange Zahl, länger als ULONG_MAX                                                                                                    |
| ERR_WRONG_FORMATSTRING      | 5037 | Falsche Formatzeile                                                                                                                    |
| ERR_TOO_MANY_FORMATTERS     | 5038 | Zahl der Formatspezifikatoren ist mehr, als Parameterzahl                                                                              |
| ERR_TOO_MANY_PARAMETERS     | 5039 | Parameterzahl ist mehr als Zahl der Formatspezifikatoren                                                                               |
| ERR_WRONG_STRING_PARAMETER  | 5040 | Beschädigter Parameter des Typs                                                                                                        |

|                                |      |                                                                                                                                      |
|--------------------------------|------|--------------------------------------------------------------------------------------------------------------------------------------|
| ERR_STRINGPOS_OUTOFRANGE       | 5041 | Position ausserhalb der Zeile                                                                                                        |
| ERR_STRING_ZEROADDED           | 5042 | Zum Zeilenende ist 0 zugefügt, sinnlose Operation                                                                                    |
| ERR_STRING_UNKNOWNTYPE         | 5043 | Unbekannter Datentyp bei Umwandlung in die Zeile                                                                                     |
| ERR_WRONG_STRING_OBJECT        | 5044 | Beschädigtes Zeilenobjekt                                                                                                            |
| <b>Operationen mit Feldern</b> |      |                                                                                                                                      |
| ERR_INCOMPATIBLE_ARRAYS        | 5050 | Kopieren der unkompatibeler Felder. Zeilenfeld kann nur in ein Zeilenfeld kopiert werden, numerisches Feld- in ein numerisches Feld. |
| ERR_SMALL_ASERIES_ARRAY        | 5051 | Aufnahmefeld ist als AS_SERIES erklärt, und es ist von der unzureichender Größe                                                      |
| ERR_SMALL_ARRAY                | 5052 | Zu kleines Feld, Startposition ausserhalb des Feldes                                                                                 |
| ERR_ZEROSIZE_ARRAY             | 5053 | Feld der Nullgröße                                                                                                                   |
| ERR_NUMBER_ARRAYS_ONLY         | 5054 | Muss numerisches Feld sein                                                                                                           |
| ERR_ONEDIM_ARRAYS_ONLY         | 5055 | Muss eindimensionales Feld sein                                                                                                      |
| ERR_SERIES_ARRAY               | 5056 | Timeserie kann nicht verwendet werden                                                                                                |
| ERR_DOUBLE_ARRAY_ONLY          | 5057 | Muss das Feld des Typs double sein                                                                                                   |
| ERR_FLOAT_ARRAY_ONLY           | 5058 | Muss das Feld des Typs float sein                                                                                                    |
| ERR_LONG_ARRAY_ONLY            | 5059 | Muss das Feld des Typs long sein                                                                                                     |
| ERR_INT_ARRAY_ONLY             | 5060 | Muss das Feld des Typs int sein                                                                                                      |
| ERR_SHORT_ARRAY_ONLY           | 5061 | Muss das Feld des Typs short sein                                                                                                    |
| ERR_CHAR_ARRAY_ONLY            | 5062 | Muss das Feld des Typs char sein                                                                                                     |
| <b>Arbeit mit OpenCL</b>       |      |                                                                                                                                      |
| ERR_OPENCL_NOT_SUPPORTED       | 5100 | <a href="#">OpenCL-Funktionen</a> werden auf diesem Computer nicht unterstützt                                                       |

|                                  |       |                                                                         |
|----------------------------------|-------|-------------------------------------------------------------------------|
| ERR_OPENCL_INTERNAL              | 5101  | Interner Fehler bei der <a href="#">Ausführung von OpenCL</a>           |
| ERR_OPENCL_INVALID_HANDLE        | 5102  | Ungültiges <a href="#">OpenCL-Handle</a>                                |
| ERR_OPENCL_CONTEXT_CREATE        | 5103  | Fehler beim Erstellen von <a href="#">OpenCL-Kontext</a>                |
| ERR_OPENCL_QUEUE_CREATE          | 5104  | Es konnte kein Laufwarteschlange in OpenCL erstellen                    |
| ERR_OPENCL_PROGRAM_CREATE        | 5105  | Fehler beim <a href="#">Kompilieren eines OpenCL-Programms</a>          |
| ERR_OPENCL_TOO_LONG_KERNEL_NAME  | 5106  | Zu lange Kernel-Name( <a href="#">OpenCL-Kernel</a> )                   |
| ERR_OPENCL_KERNEL_CREATE         | 5107  | Fehler beim Erstellen vom <a href="#">OpenCL-Kernel</a>                 |
| ERR_OPENCL_SET_KERNEL_PARAMETER  | 5108  | Fehler bei <a href="#">Eingabe von Parameters</a> für den OpenCL-Kernel |
| ERR_OPENCL_EXECUTE               | 5109  | Laufzeitfehler vom <a href="#">OpenCL-Programm</a>                      |
| ERR_OPENCL_WRONG_BUFFER_SIZE     | 5110  | Ungültige Größe des OpenCL-Puffers                                      |
| ERR_OPENCL_WRONG_BUFFER_OFFSET   | 5111  | Ungültiges Offset im OpenCL-Puffer                                      |
| ERR_OPENCL_BUFFER_CREATE         | 5112  | Fehler beim Erstellen des <a href="#">OpenCL-Puffers</a>                |
| <b>Работа с WebRequet</b>        |       |                                                                         |
| ERR_WEBREQUEST_INVALID_ADDRESS   | 5200  | URL не прошел проверку                                                  |
| ERR_WEBREQUEST_CONNECT_FAILED    | 5201  | Не удалось подключиться к указанному URL                                |
| ERR_WEBREQUEST_TIMEOUT           | 5202  | Превышен таймаут получения данных                                       |
| ERR_WEBREQUEST_REQUEST_FAILED    | 5203  | Ошибка в результате выполнения HTTP запроса                             |
| <b>Benutzerdefinierte Fehler</b> |       |                                                                         |
| ERR_USER_ERROR_FIRST             | 65536 | Mit diesem Kode fangen <a href="#">benutzerdefinierte Fehler</a> an     |

Sehen Sie auch

Rückgabecodes des Handelsservers

## Eingabe/Ausgabe Konstanten

Konstanten:

- [Flaggen der Dateieröffnung](#)
- [Dateieigenschaften](#)
- [Positionieren innerhalb der Datei](#)
- [Kodeseite Verwenden](#)
- [MessageBox](#)

## Flaggen der Dateieröffnung

Werte von Flaggen, die das Regime von Arbeit mit Dateien bestimmen. Flaggen werden folgenderweise definiert:

| Identifikator | Wert | Beschreibung                                                                                                                                                                                                                                    |
|---------------|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| FILE_READ     | 1    | Datei wird für Lesen geöffnet. Flagge wird beim Öffnen von Dateien ( <a href="#">FileOpen()</a> ) verwendet. Bei der Öffnung der Datei muss unbedingt die Flagge FILE_WRITE und/oder die Flagge FILE_READ angegeben wird.                       |
| FILE_WRITE    | 2    | Datei wird für Schreiben verwendet. Flagge wird beim Öffnen von Dateien ( <a href="#">FileOpen()</a> ) verwendet. Bei der Öffnung der Datei muss unbedingt die Flagge FILE_WRITE und/oder die Flagge FILE_READ angegeben wird.                  |
| FILE_BIN      | 4    | Binäres Regime von Lesen-Schreiben (ohne Umwandlung von Zeile in die Zeile). Flagge wird beim Öffnen von Dateien ( <a href="#">FileOpen()</a> ) verwendet                                                                                       |
| FILE_CSV      | 8    | Datei des Typs csv (alle geschriebenen Elemente werden in die Zeilen des entsprechenden Typs verwendet, unicode oder ansi, und durch Begrenzungszeichen getrennt). Flagge wird beim Öffnen von Dateien ( <a href="#">FileOpen()</a> ) verwendet |
| FILE_TXT      | 16   | Einfache Textdatei (ebensolcher wie csv, aber der Begrenzungszeichen wird nicht beachtet). Flagge wird beim Öffnen von Dateien ( <a href="#">FileOpen()</a> ) verwendet                                                                         |
| FILE_ANSI     | 32   | Zeilen des Typs ANSI (1-Byte Symbole). Flagge wird beim Öffnen von Dateien ( <a href="#">FileOpen()</a> ) verwendet                                                                                                                             |

|                  |      |                                                                                                                                                                                                                                                                                                 |
|------------------|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| FILE_UNICODE     | 64   | Zeilen des Typs UNICODE (2-Byte-Symbole). Flagge wird beim Öffnen von Dateien ( <a href="#">FileOpen()</a> ) verwendet                                                                                                                                                                          |
| FILE_SHARE_READ  | 128  | Gemeinsamer Lesezugriff von mehreren Programmen. Flagge wird beim Öffnen von Dateien ( <a href="#">FileOpen()</a> ) verwendet, aber ersetzt es nicht die Notwendigkeit, beim Dateieröffnung FILE_WRITE und/oder FILE_READ anzugeben                                                             |
| FILE_SHARE_WRITE | 256  | Gemeinsamer Schreibzugriff von mehreren Programmen. Flagge wird beim Öffnen von Dateien ( <a href="#">FileOpen()</a> ) verwendet, aber ersetzt es nicht die Notwendigkeit, beim Dateieröffnung FILE_WRITE und/oder FILE_READ anzugeben                                                          |
| FILE_REWRITE     | 512  | Möglichkeit der Dateispeicherung durch die Funktionen <a href="#">FileCopy()</a> und/oder <a href="#">FileMove()</a> . Datei muss vorhanden sein oder für Schreiben geöffnet werden. Anders wird die Datei nicht geöffnet werden                                                                |
| FILE_COMMON      | 4096 | Datei /// im Verzeichnis aller Kundenterminals \Terminal\Common\Files. Flagge wird beim Öffnen von Dateien ( <a href="#">FileOpen()</a> ), Dateikopieren ( <a href="#">FileCopy()</a> ), <a href="#">FileMove()</a> und Prüfung von Dateianwesenheit ( <a href="#">FileExists()</a> ) verwendet |

Bei der Öffnung der Datei kann eine oder mehrere Flaggen angegeben werden, diese Verbindung heißt Kombination der Flaggen. Kombination der Flaggen wird mittels des Zeichens der Operation des logischen ODER (|) geschrieben, das zwischen den aufgezählten Flaggen gestellt wird. ZB um eine Datei in CSV Format für Lesen und Schreiben zu öffnen, kann die Kombination FILE\_READ|FILE\_WRITE|FILE\_CSV angegeben werden.

#### Beispiel:

```
int filehandle=FileOpen(filename,FILE_READ|FILE_WRITE|FILE_CSV);
```

Es gibt einige Besonderheiten der Arbeit bei Andeutung der Flaggen des Lesens und der Schreibung:

- Wenn FILE\_READ angegeben wird, versucht man, die schon existierende Datei zu öffnen. Wenn die Datei existiert, gelingt es nicht die Datei zu öffnen, eine neue Datei wird nicht erzeugt.
- Wenn FILE\_READ|FILE\_WRITE - eine neue Datei wird erzeugt, wenn die Datei mit diesem Namen fehlt.
- Wenn FILE\_WRITE - Datei wird mit der Null-Grösse erzeugt.

Bei der Öffnung der Datei muss unbedingt die Flagge FILE\_WRITE und/oder die Flagge FILE\_READ angegeben werden.

Flaggen, die den Typ der offenen Datei bestimmen, haben die Priorität. Die älteste Flagge ist FILE\_CSV, dann folgt die zweitälteste Flagge FILE\_BIN, und die kleinste Priorität hat die Flagge FILE\_TXT. So wenn mehrere Dateien gleichzeitig (FILE\_TXT|FILE\_CSV oder FILE\_TXT|FILE\_BIN oder FILE\_BIN|FILE\_CSV) angegeben werden, wird die älteste Flagge verwendet werden.

Flaggen, die den Typ der Kodierung bestimmen, haben auch eine Priorität. Flagge FILE\_UNICODE hat die ältere Priorität als die Flagge FILE\_ANSI. Darum wird bei der Andeutung der Kombination FILE\_UNICODE|FILE\_ANSI die Flagge FILE\_UNICODE verwendet.

Wenn FILE\_UNICODE, und FILE\_ANSI angegeben wird, wird FILE\_UNICODE gemeint. Wenn FILE\_CSV, und FILE\_BIN nicht angegeben wird, und FILE\_TXT, wird FILE\_CSV gemeint

Wenn die Datei als Textdatei für Lesen geöffnet wird (FILE\_TXT oder FILE\_CSV), und dabei wird am Anfang der Datei ein spezielles Zwei-Byte Zeichen **0xff,0xfe gefunden**, wird Flagge der Kodierung FILE\_UNICODE sein, auch wenn die Flagge FILE\_ANSI angegeben wurde.

Sehen Sie auch

[Dateioperationen](#)



## Dateieigenschaften

Die Funktion [FileGetInteger\(\)](#), ist für den Erhalt von Dateieigenschaften verwendet. Der Identifier der gewünschten Eigenschaft ist dieser Funktion aus `ENUM_FILE_PROPERTY_INTEGER` beim Aufruf übergeben.

### ENUM\_FILE\_PROPERTY\_INTEGER

| ID               | ID Beschreibung                                                                                                    |
|------------------|--------------------------------------------------------------------------------------------------------------------|
| FILE_EXISTS      | Prüfung der Existenz                                                                                               |
| FILE_CREATE_DATE | Erstellungsdatum                                                                                                   |
| FILE_MODIFY_DATE | Datum der letzten Überarbeitung                                                                                    |
| FILE_ACCESS_DATE | Datum der letzten Zugriff zur Datei                                                                                |
| FILE_SIZE        | Dateigröße in Byte                                                                                                 |
| FILE_POSITION    | Position eines Zeigers in der Datei                                                                                |
| FILE_END         | Zeichen der Dateiende erhalten                                                                                     |
| FILE_LINE_END    | Zeichen der Zeilenende erhalten                                                                                    |
| FILE_IS_COMMON   | Die Datei wird in einem freigegebenen Ordner für allen Terminals geöffnet (Sehen Sie <a href="#">FILE_COMMON</a> ) |
| FILE_IS_TEXT     | Die Datei ist als eine Textdatei geöffnet (Sehen Sie <a href="#">FILE_TXT</a> )                                    |
| FILE_IS_BINARY   | Die Datei ist als eine binäre Datei geöffnet (Sehen Sie <a href="#">FILE_BIN</a> )                                 |
| FILE_IS_CSV      | Die Datei ist als CSV geöffnet (Sehen Sie <a href="#">FILE_CSV</a> )                                               |
| FILE_IS_ANSI     | Die Datei ist als ANSI geöffnet (Sehen Sie <a href="#">FILE_ANSI</a> )                                             |
| FILE_IS_READABLE | Die geöffnete Datei ist lesbar (Sehen Sie <a href="#">FILE_READ</a> )                                              |
| FILE_IS_WRITABLE | Die geöffnete Datei ist beschreibbar (Sehen Sie <a href="#">FILE_WRITE</a> )                                       |

Die Funktion [FileGetInteger\(\)](#) hat zwei Varianten der Abruf In der ersten Version, um die Eigenschaften der Datei zu erhalten, eingeben Sie ihr Handle, der beim Öffnen Der Datei mit Funktion [FileOpen\(\)](#) erhalten war. Diese Version ermöglicht es Ihnen, alle Dateieigenschaften zu bekommen.

Die zweite Version der Funktion [FileGetInteger\(\)](#) gibt Werte der Dateieigenschaften durch ihren Namen zurück. In dieser Version können Sie nur die folgende allgemeine Eigenschaften erhalten:

- `FILE_EXISTS` - ob die Datei mit dem eingegebenen Namen existiert;
- `FILE_CREATE_DATE` - Erstellungsdatum der Datei mit dem eingegebenen Namen;

- FILE\_MODIFY\_DATE - Datum der letzten Überarbeitung der Datei mit dem eingegebenen Namen;
- FILE\_ACCESS\_DATE - Datum der letzten Zugriff zur Datei mit dem eingegebenen Namen;
- FILE\_SIZE - Größe der Datei mit dem eingegebenen Namen.

Wenn Sie andere Eigenschaften als die oben genannten zu erhalten versuchen, gibt die zweite Version der FileGetInteger() einen Fehler zurück.

## Positionieren innerhalb der Dateien

Die meisten [Dateifunktionen](#) sind mit den Funktionen von Lesen/Aufzeichnung der Information verbunden. Dabei kann durch die Funktion [FileSeek\(\)](#) der Dateianzeiger in der Position innerhalb der Datei angegeben werden, von der die nächste Lese- oder Schreiboperation durchgeführt wird. Aufzählung `ENUM_FILE_POSITION` enthält zulaessige Anzeigerpositionen, in Bezug auf die Verschiebungen in Bytes für nächste Operation spezifiziert werden können.

### ENUM\_FILE\_POSITION

| Identifikator | Beschreibung                         |
|---------------|--------------------------------------|
| SEEK_SET      | Dateianfang                          |
| SEEK_CUR      | Laufende Position des Dateianzeigers |
| SEEK_END      | Dateiende                            |

Sehen Sie auch

[FileIsEnding](#), [FileIsLineEnding](#)

## Kodeseite Verwenden in Zeilenumwandlung

Beim Konvertieren der [Zeilenvariablen](#) in Felder des Typs [char](#) und umgekehrt wird in der Sprache MQL5 Kodieren verwendet, das dem laufenden ANSI Kodieren des Operationssystems Windows (CP\_ACP) entspricht. Wenn man braucht, einen anderen Kodierungstyp anzugeben, kann man es durch Zusatzparameter für Funktionen [CharArrayToString\(\)](#), [StringToCharArray\(\)](#) und [FileOpen\(\)](#) einstellen.

In der Tabelle werden eingebaute Konstanten für einige am meisten angeforderte Kodeseiten angegeben. Nicht angegebene Kodeseiten können mit Hilfe des Codes angegeben werden, der dieser Seite entspricht.

### Eingebaute Konstanten der Kodeseiten

| Konstante     | Wert  | Beschreibung                                                                                                                                                                                                    |
|---------------|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CP_ACP        | 0     | Laufende Kodeseite ANSIm Operationssystem Windows                                                                                                                                                               |
| CP_OEMCP      | 1     | Laufende Kodeseite OEM.                                                                                                                                                                                         |
| CP_MACP       | 2     | Laufende Kodeseite Macintosh.<br><b>Bemerkung:</b> Dieser Wert wird vor allem in früher erzeugten Programmcodes verwendet und ist nicht notwendig, denn moderne Computers Macintosh verwenden Unicode Kodieren. |
| CP_THREAD_ACP | 3     | Kodieren Windows ANSI für laufenden Thread.                                                                                                                                                                     |
| CP_SYMBOL     | 42    | Kodeseite Symbol                                                                                                                                                                                                |
| CP_UTF7       | 65000 | Kodeseite UTF-7.                                                                                                                                                                                                |
| CP_UTF8       | 65001 | Kodeseite UTF-8.                                                                                                                                                                                                |

Sehen Sie auch

[Zustand von Client Terminal](#)

## Konstanten des Dialogfensters MessageBox

Rückgabekodes der Funktion [MessageBox\(\)](#). Wenn Message window Schaltfläche Cancel hat, gibt die Funktion den Wert IDCANCEL bei der gedrückten Taste ESC oder Schaltfläche Cancel zurück. Wenn message window keine Schaltfläche Cancel hat, wird das Drücken ESC keinen Effekt haben.

| Konstante  | Wert | Beschreibung                           |
|------------|------|----------------------------------------|
| IDOK       | 1    | Schaltfläche OK wird ausgewählt        |
| IDCANCEL   | 2    | Schaltfläche Cancel wird ausgewählt    |
| IDABORT    | 3    | Schaltfläche Abort wird ausgewählt     |
| IDRETRY    | 4    | Schaltfläche Retry wird ausgewählt     |
| IDIGNORE   | 5    | Schaltfläche Ignore wird ausgewählt    |
| IDYES      | 6    | Schaltfläche Yes wird ausgewählt       |
| IDNO       | 7    | Schaltfläche No wird ausgewählt        |
| IDTRYAGAIN | 10   | Schaltfläche Try Again wird ausgewählt |
| IDCONTINUE | 11   | Schaltfläche Continue wird ausgewählt  |

Die Hauptflaggen der Funktion [MessageBox\(\)](#) bestimmen den Inhalt und das Verhalten des Dialogfensters. Dieser Wert kann Flaggenkombination aus folgenden Flagengruppen sein:

| Konstante           | Wert       | Beschreibung                                                   |
|---------------------|------------|----------------------------------------------------------------|
| MB_OK               | 0x00000000 | Message Window hat eine Schaltfläche: OK. Default              |
| MB_OKCANCEL         | 0x00000001 | Message Window hat zwei Schaltflächen: OK und Cancel           |
| MB_ABORTRETRYIGNORE | 0x00000002 | Message Window hat drei Schaltflächen: Abort, Retry und Ignore |
| MB_YESNOCANCEL      | 0x00000003 | Message Window hat drei Schaltflächen: Yes, No und Cancel      |

|                      |            |                                                                    |
|----------------------|------------|--------------------------------------------------------------------|
| MB_YESNO             | 0x00000004 | Message Window hat zwei Schaltflächen: Yes und No                  |
| MB_RETRYCANCEL       | 0x00000005 | Message Window hat zwei Schaltflächen: Retry und Cancel            |
| MB_CANCELTRYCONTINUE | 0x00000006 | Message Window hat drei Schaltflächen: Cancel, Try Again, Continue |

für Darstellung der Ikone muss man in Message Window zusätzliche Flaggen spezifiziert werden:

| Konstante                                    | Wert       | Beschreibung                                       |
|----------------------------------------------|------------|----------------------------------------------------|
| MB_ICONSTOP,<br>MB_ICONERROR,<br>MB_ICONHAND | 0x00000010 | Darstellung von STOP Zeichen                       |
| MB_ICONQUESTION                              | 0x00000020 | Darstellung von Fragezeichen                       |
| MB_ICONEXCLAMATION,<br>MB_ICONWARNING        | 0x00000030 | Darstellung von Aufrufzeichen                      |
| MB_ICONINFORMATION,<br>MB_ICONASTERISK       | 0x00000040 | Darstellung aus dem eingekreisten Zeichen <b>i</b> |

Default -0 Schaltfläche werden durch folgende Flaggen vorgegeben:

| Konstante     | Wert       | Beschreibung                                                                                                                        |
|---------------|------------|-------------------------------------------------------------------------------------------------------------------------------------|
| MB_DEFBUTTON1 | 0x00000000 | Die erste Schaltfläche MB_DEFBUTTON1 - ist Default, wenn MB_DEFBUTTON2, MB_DEFBUTTON3, oder MB_DEFBUTTON4 nicht spezifiziert werden |
| MB_DEFBUTTON2 | 0x00000100 | zweite Schaltfläche - Default-Wert                                                                                                  |
| MB_DEFBUTTON3 | 0x00000200 | dritte Schaltfläche - Default-Wert                                                                                                  |
| MB_DEFBUTTON4 | 0x00000300 | Vierte Schaltfläche - Default-Wert                                                                                                  |

## Programme MQL5

für die Arbeit des mql5-Programms muss es kompiliert werden (Druckknopf "Kompilieren" oder Taste F7). Compilierung muss ohne Fehler vergehen (zugelassen sind Warnungen, die analysiert werden müssen). Dabei muss im entsprechenden Verzeichnis, *terminal\_dir\MQL5\Experts*, *terminal\_dir\MQL5\indicators* oder *terminal\_dir\MQL5\scripts*, Durchführungsdatei mit demselben Namen und Erweiterung EX5 erzeugt werden. Gerade diese Datei muss abgelaufen lassen.

Besonderheiten der mql5-Programme werden in folgenden Abschnitten beschrieben:

- [Programmausführung](#) - Verfahren des Aufrufs der vorbestimmten Funktionen-Bearbeiter der Ereignisse;
- [Testen von Handelsstrategien](#) - Besonderheiten von Programmen im Strategie-Tester;
- [Ereignisse des Kundenterminals](#) - Beschreibung der Ereignisse, die in Programmen verarbeitet werden können;
- [Aufruf der importierten Funktionen](#) - Verfahren der Beschreibung, zulaessige Parameter, Suchedaten und Abkommen der importierten Funktionen;
- [Durchführungsfehler](#) - Erhalten der Information über Durchführungsfehler und kritische Fehler.

Experten, Benutzeranzeiger und Scripts werden zum einen der offenen Charts durch Drag'n'Drop aus dem Fenster "Navigator" des Kundenterminals auf entsprechenden Chart angehängt. Mql5-Programme können nur beim angeschalteten Kundenterminals funktionieren.

Damit Expert zu arbeiten stoppt, muss er vom Chart durch die Wahl "Ratgeber - Entfernen" vom Kontextmenue des Charts entfernt werden. Arbeit der Ratgeber werden auch vom Verhalten des Knopfes "Ratgeber Erlauben/verbieten" beeinflusst.

Damit Benutzeranzeiger zu arbeiten stoppt, muss man ihn vom Chart entfernen.

Benutzeranzeiger und Ratgeber arbeiten bis sie explizit vom Chart entfernt werden; Information über angehängte Ratgeber und Benutzeranzeiger wird zwischen Starts des Kundenterminals gespeichert.

Scripts werden einmal ausgeführt und nach Arbeitsabschluss oder bei Veränderung des Verhaltens des laufenden Charts oder nach Abschluss des Kundenterminals automatisch entfernt. Beim neuen Start des Kundenterminals werden Scripts nicht ablaufen lassen, denn die Information über sie wird nicht gespeichert.

Auf einem Chart können ein Expert, ein Script und unbegrenzte Zahl der Anzeiger funktionieren.

## Durchführung der Programme

Jedes Skript und jeder Expert Advisor läuft in einem eigenen separaten Thread. Alle Indikatoren, die auf ein Symbol berechnet sind, auch wenn sie auf unterschiedlichen Charts laufen, laufen im selben Thread. Somit benutzen Indikatoren eines Symbols die Ressourcen eines einzigen Threads gemeinsam.

Alle anderen Aktionen, die mit einem Symbol verbunden sind, wie die Verarbeitung von Ticks und die Synchronisation der Geschichte werden auch konsequent in dem selben Thread mit Indikatoren durchgeführt. Dies bedeutet, dass, wenn der Indikator hält unendliche Aktion, alle anderen Ereignisse in diesem Symbol nie durchgeführt werden.

Beim Starten eines Expert Advisors, stellen Sie sicher dass er aktuelle [Handelsumwelt](#) hat und [Geschichte](#) des Symbols für die Periode, und [synchronisieren](#) Sie Daten zwischen dem Terminal und dem Server. Für alle diese Verfahren stellt das Terminal eine Startverzögerung von nicht mehr als 5 Sekunden, wonach der Expert Advisor mit verfügbaren Daten gestartet wird. Daher, wenn es kein Verbindung zum Server gibt, kann dies zu einer Verzögerung des Starts des Expert Advisors führen.

Eine kurze Zusammenfassung des MQL5-Programmen ist in der Tabelle dargestellt:

| Programm       | Durchführung                                                                                                           | Bemerkung                                                                                                       |
|----------------|------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------|
| Skript         | In einem eigenen Thread, wie viele Skripte, so viele Ausführungsthreads für sie                                        | Ein endloses Skript kann nicht Ausführung andere Programme brechen                                              |
| Expert Advisor | In einem eigenen Thread, wie viele Expert Advisors, so viele Ausführungsthreads für sie                                | Ein endloser Expert Advisor nicht Ausführung andere Programme brechen                                           |
| Indikator      | Ein Thread für alle Indikatoren auf ein Symbol. Wie viele Symbole mit Indikatoren, so viele Ausführungsthreads für sie | Eine Endlosschleife in einem des Indicators wird Ausführung aller anderen Indikatoren auf diesem Symbol stoppen |

Sofort nach Anhängen des Programms zum Chart Erfolgt sein Einspeichern in Kundenterminal und in Speicher des Kundenterminals und [Initialisierung](#) der globalen Variablen. Wenn eine der globalen Variablen des Typen Klasse [Konstrukteur](#) hat, wird dieser Konstrukteur während der Initialisierung der [globalen Variablen](#) aufgerufen werden.

Danach wartet das Programm auf das [Ereignis](#) vom Kundenterminal. Jedes mql5-Programm muss wenigstens eine [Funktion-Bearbeiter](#) des Ereignisses haben, sonst wird das Programm nicht durchgeführt werden. Funktionen-Bearbeiter haben vorbestimmte Namen, vorbestimmte Parametervorraete und vorbestimmte Rückkehrtypen.

| Typ | Funktionsname          | Parameter | Anwendung             | Bemerkung                                                           |
|-----|------------------------|-----------|-----------------------|---------------------------------------------------------------------|
| int | <a href="#">OnInit</a> | nicht     | Experten und Anzeiger | Bearbeiter des Ereignisses <a href="#">Init.</a> Zugelassen Typ des |



|      |                             |                                                                                                                                                                                                                                                                                                |                          |                                                                                                                                                                                                                                                                          |
|------|-----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|      |                             |                                                                                                                                                                                                                                                                                                |                          | Rückgabewertes<br>void.                                                                                                                                                                                                                                                  |
| void | <a href="#">OnDeinit</a>    | const int reason                                                                                                                                                                                                                                                                               | Experten und<br>Anzeiger | Bearbeiter des<br>Ereignisses<br><a href="#">Deinit</a> .                                                                                                                                                                                                                |
| void | <a href="#">OnStart</a>     | nicht                                                                                                                                                                                                                                                                                          | Scripts                  | Bearbeiter des<br>Ereignisses<br><a href="#">Start</a> .                                                                                                                                                                                                                 |
| int  | <a href="#">OnCalculate</a> | const int<br>rates_total,<br>const int<br>prev_calculated,<br>const datetime<br>&Time[],<br>const double<br>&Open[],<br>const double<br>&High[],<br>const double<br>&Low[],<br>const double<br>&Close[],<br>const long<br>&TickVolume[],<br>const long<br>&Volume[],<br>const int<br>&Spread[] | Anzeiger                 | Bearbeiter des<br>Ereignisses<br>Calculate für alle<br>Preisdaten.                                                                                                                                                                                                       |
| int  | <a href="#">OnCalculate</a> | const int<br>rates_total,<br>const int<br>prev_calculated,<br>const int begin,<br>const double<br>&price[]                                                                                                                                                                                     | Anzeiger                 | Ereignisbearbeit<br>er <a href="#">Calculate</a> in<br>einem Datenfeld.<br>In Indikator<br>können nicht 2<br>Bearbeiter<br>Calculate<br>gleichzeitig sein.<br>In diesem Fall<br>wird nur ein<br>Ereignisbearbeit<br>er Calculate in<br>einem Datenfeld<br>funktionieren. |
| void | <a href="#">OnTick</a>      | nicht                                                                                                                                                                                                                                                                                          | Experten                 | Ereignisbearbeit<br>er <a href="#">NewTick</a> .<br>Wenn das<br>Ereignis des<br>neuen Ticks                                                                                                                                                                              |

|        |                              |                                                                                                |                          |                                                                           |
|--------|------------------------------|------------------------------------------------------------------------------------------------|--------------------------|---------------------------------------------------------------------------|
|        |                              |                                                                                                |                          | verarbeitet wird, werden andere Ereignisse dieses Typs nicht aufgenommen. |
| void   | <a href="#">OnTimer</a>      | nicht                                                                                          | Experten und Anzeiger    | Ereignisbearbeit er <a href="#">Timer</a> .                               |
| void   | <a href="#">OnTrade</a>      | nicht                                                                                          | Experten                 | Ereignisbearbeit er <a href="#">Trade</a> .                               |
| double | <a href="#">OnTester</a>     | nicht                                                                                          | Experten                 | Ereignisbearbeit er <a href="#">Tester</a> .                              |
| void   | <a href="#">OnChartEvent</a> | const int id,<br>const long<br>&lparam,<br>const double<br>&dparam,<br>const string<br>&sparam | Experten und Indikatoren | Ereignisbearbeit er <a href="#">ChartEvent</a> .                          |
| void   | <a href="#">OnBookEvent</a>  | const string<br>&symbol_name                                                                   | Experten und Indikatoren | Ereignisbearbeit er <a href="#">BookEvent</a> .                           |

Der Kundenterminal sendet die neuen Ereignisse in die entsprechende offene Charts. Darüber hinaus können die Ereignisse durch Charts ([Chartseignisse](#)) oder MQL5-Programme ([benutzerdefinierte Ereignisse](#)) generiert werden. Generierung von Ereignisse der Erstellung oder Löschen von graphischen Objekten auf einen Chart kann durch [CHART\\_EVENT\\_OBJECT\\_CREATE](#) und [CHART\\_EVENT\\_OBJECT\\_DELETE](#) Chartseigenschaften aktiviert oder deaktiviert werden. Jedes MQL5-Programm und jeder Chart hat eine eigene Warteschlange von Ereignissen, in die alle neu eingehenden Ereignisse hinzugefügt werden.

Ein Programm erhält Ereignisse nur aus dem Chart, auf dem es läuft. Alle Ereignisse werden nacheinander in der Reihenfolge des Eingangs bearbeitet. Wenn eine Warteschlange bereits eine [NewTick](#) Ereignis hat, oder dieses Ereignis ist gerade in Bearbeitung, dann wird nicht die neue NewTick Ereignis in der Warteschlange des MQL5 Programms platziert. Auch wenn [ChartEvent](#) bereits in der Warteschlange ist, oder dieses Ereignis in Bearbeitung ist, wird kein neues Ereignis dieser Art eingerichtet werden. Die Timer-Ereignisse werden auf die gleiche Weise behandelt - wenn ein [Timer](#)-Ereignis in der Warteschlange ist oder behandelt wird, die neue Timer-Ereignis wird nicht in der Warteschlange platziert.

Ereignis-Warteschlangen haben eine begrenzte, aber ausreichende Größe, so dass die Warteschlange-Überlauf für gut geschriebene Programme unwahrscheinlich ist. Im Falle einer Warteschlange-Überlauf, werden neue Ereignisse ohne Wartezeiten verworfen.

Es wird nicht empfohlen, um Endlosschleifen zu verwenden, um Ereignisse zu verarbeiten. Die Ausnahme zu dieser Regel sind nur Skripte, die nur eine einzige [Start](#)-Ereignis verarbeiten.

[Bibliotheken](#) verarbeiten keine Ereignisse.

## Das Verbot der Verwendung von Funktionen in Indikatoren und Experten

Indikatoren, Skripte und Expert Advisors sind in MQL5 geschriebene ausführbare Programme. Sie sind für verschiedene Arten von Aufgaben ausgelegt. Deshalb gibt es einige Beschränkungen für die Verwendung bestimmter Funktionen, abhängig von der [Art des Programms](#). Die folgenden Funktionen werden in Indikatoren verboten:

- [OrderCalcMargin\(\)](#);
- [OrderCalcProfit\(\)](#);
- [OrderCheck\(\)](#);
- [OrderSend\(\)](#);
- [SendFTP\(\)](#);
- [Sleep\(\)](#);
- [ExpertRemove\(\)](#);
- [MessageBox\(\)](#).

Alle Funktionen für Indikatoren sind in Expert Advisors und Skripte verboten:

- [SetIndexBuffer\(\)](#);
- [IndicatorSetDouble\(\)](#);
- [IndicatorSetInteger\(\)](#);
- [IndicatorSetString\(\)](#);
- [PlotIndexSetDouble\(\)](#);
- [PlotIndexSetInteger\(\)](#);
- [PlotIndexSetString\(\)](#);
- [PlotIndexGetInteger](#).

Die Bibliothek ist kein eigenständiges Programm und wird im Rahmen des Programms MQL5, die sie aufgerufen hat, ausgeführt: Skript, Indikator oder Expert Advisor. Dementsprechend werden die oben genannten Beschränkungen für die genannte Bibliothek angewendet.

## Laden und Ausladen der Indikatoren

Indikatoren werden in folgenden Fällen ausgeladen:

- Anhängen des Indikators zum Chart;
- Start des Terminals (wenn Indikator vor dem früheren Terminalschliessen zum Chart angehängt wurde);
- Laden der Schablone (wenn in der Schablone Indikator angegeben wird, angehängt zum Chart);
- Profilwechsel (wenn Indikator zum einen der Profilcharts angehängt wird);
- Wechsel des Symbols und/oder Chartperiode, zu dem der Indikator angehängt wird;
- nach erfolgreicher Neucompilierung des Indikators, wenn dieser Anzeiger zum Chart angehängt wurde.

- Wechsel der [Eingabeparameter](#) des Indikators.

Indikatoren werden in folgenden Fällen ausgeladen:

- bei Abtrennung des Indikators vom Chart;
- Terminalschiessen (wenn Indikator zum Chart angehängt wurde);
- Laden der Schablone, Indikator zum Chart angehängt wird;
- Schliessen des Charts, zu dem Indikator angehängt wurde;
- Profilwechsel, wenn Indikator zu einem der Charts des Wechselprofils angehängt wird;
- Wechsel des Symbols und/oder Chartperiode, zu dem Indikator angehängt wird;
- Wechsel der Eingabeparameter des Indikators.

## Laden und Ausladen der Experte

Laden des Experten erfolgt in folgenden Fällen:

- Anhängen von Experten zum Chart;
- Start des Terminals (wenn Expert vor dem früheren Terminalabschluss zum Chart angehängt wurde);
- Laden der Schablone (wenn in der Schablone Expert angegeben wird, angehängt zum Chart);
- nach erfolgreicher Neucompilierung des Experten, wenn dieser Expert zum Chart angehängt wurde.
- Profilwechsel (Wenn Expert zum einen der Profilcharts angehängt wird);
- Verbindung zu einem Konto, auch wenn die Kontonummer nicht geändert wurde (wenn der Expert Advisor wurde um den Chart, bevor die Zulassung des Terminals auf dem Server, verbunden).

Ausladen des Experten, angehängt zum Chart erfolgt in folgenden Fällen:

- bei Abtrennung des Experten vom Chart;
- beim Anhängen des Experten zum Chart - wenn auf dem Chart einen anderen Experten gab, wird dieser Expert ausgeladen;
- Terminalschiessen, wenn der Expert zum Chart angehängt wurde;
- Laden der Schablone, wenn der Expert zum Chart angehängt wird;
- Schliessen des Charts, zu dem Expert angehängt wurde;
- Profilwechsel, wenn der Expert zum einen der Charts der Wechselprofile angehängt wird;
- Änderung des Kontos, zu dem das Terminal verbunden ist (wenn der Expert Advisor wurde um den Chart, bevor die Zulassung des Terminals auf dem Server, verbunden).

**Beim Wechsel des Symbols oder Timeframe des Charts, zu dem Expert angehängt wird, erfolgt Laden und Ausladen des Experten nicht.** Dabei werden Bearbeiter [OnDeinit\(\)](#) auf altem Symbol/Timeframe und [OnInit\(\)](#) auf neuem Symbol/Timeframe (wenn es die solchen geben) serienweise aufgerufen, Werte der globalen Variablen und [statischer Variablen](#) werden nicht auf Null gesetzt. Alle Ereignisse, die vor dem Initialisierungsende (Funktion [OnInit\(\)](#)) erscheinen, werden weggelassen.

## Laden und Ausladen von Scripts

Scripts werden sofort nach Anhängen zum Chart geladen und sofort nach Operationsende ausgeladen. Dabei werden die Funktionen OnInit() und OnDeinit() für Scripts nicht aufgerufen.

Beim Programmausladen (Programmenfernen vom Chart) erfolgt Deinitialisierung der globalen Variablen und Entfernung der Queue von Ereignissen. In diesem Fall bedeutet Deinitialisierung Freigabe der Variablen des Typs string, Freigabe der Objekte der dynamischen Felder und Aufruf der Destruktors, wenn sie es geben.

für bessere Verständigung der Expertenarbeit ist es empfehlenswert, Kode des im Beispiel angeführten Experten zu kompilieren und Operationen von Laden/Ausladen der Experten, Schablonen-, Symbol-, Timeframewechsel usw. durchführen.

#### Beispiel:

```
//+-----+
//|                                     TestExpert.mq5 |
//|                                     Copyright 2009, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

class CTestClass
{
public:
    CTestClass() { Print("CTestClass constructor"); }
    ~CTestClass() { Print("CTestClass destructor"); }
};
CTestClass global;
//+-----+
//| Expert initialization function |
//+-----+
int OnInit()
{
    //---
    Print("Initialisation");
    //---
    return(INIT_SUCCEEDED);
}
//+-----+
//| Expert deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
    //---
    Print("Deinitialisation with reason",reason);
}
//+-----+
//| Expert tick function |
//+-----+
void OnTick()
{
    //---
}
//+-----+
```

Scripts werden sofort nach Anhängen zum Chart geladen und sofort nach Operationsende ausgeladen.

Sehen Sie auch

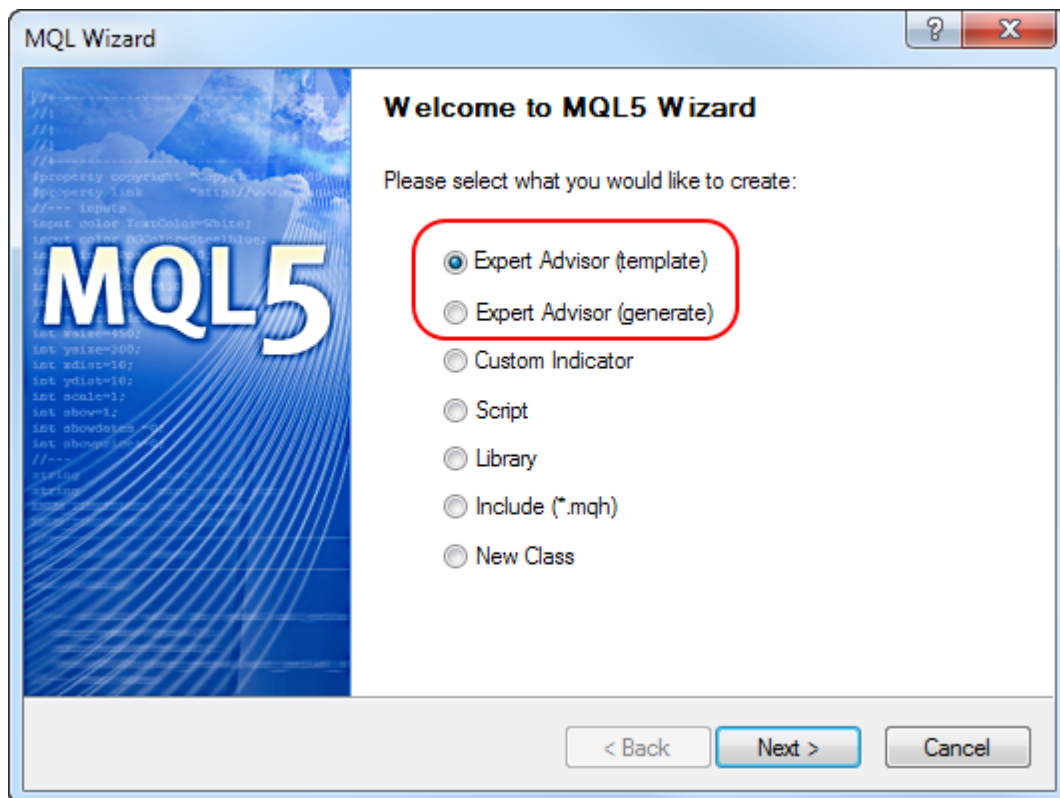
[Ereignisse des Kundenterminals](#), [Ereignisbearbeiter](#)

## Handelserlaubnis

### Handelsautomatisierung

Die MQL5 Sprache hat eine spezielle Gruppe von [Handelsfunktionen](#), mit denen man automatisierte Handelssysteme erstellen kann. Programme für den automatisierten Handel ohne menschliche Eingriffe sind Experten (Expert Advisor) oder Trading Roboter genannt. Um einen Experte in MetaEditor zu erstellen, starten sie den Assistent zur Erstellung von Experten MQL5 Wizard, und wählen Sie eine der Optionen:

- Expert Advisor (template) - ermöglicht eine Vorlage mit bereits [Event-Handling-Funktionen](#) erstellen, die soll mit allen notwendigen Funktionen durch Selbstprogrammierung ergänzt werden.
- Expert Advisor (generate) - ermöglicht [einen fertigen Trading Roboter zu erstellen](#) durch die Auswahl der Module, die Sie benötigen: Modul der Handelssignale, Geld-Management-Modul und Trailing-Stop-Modul.



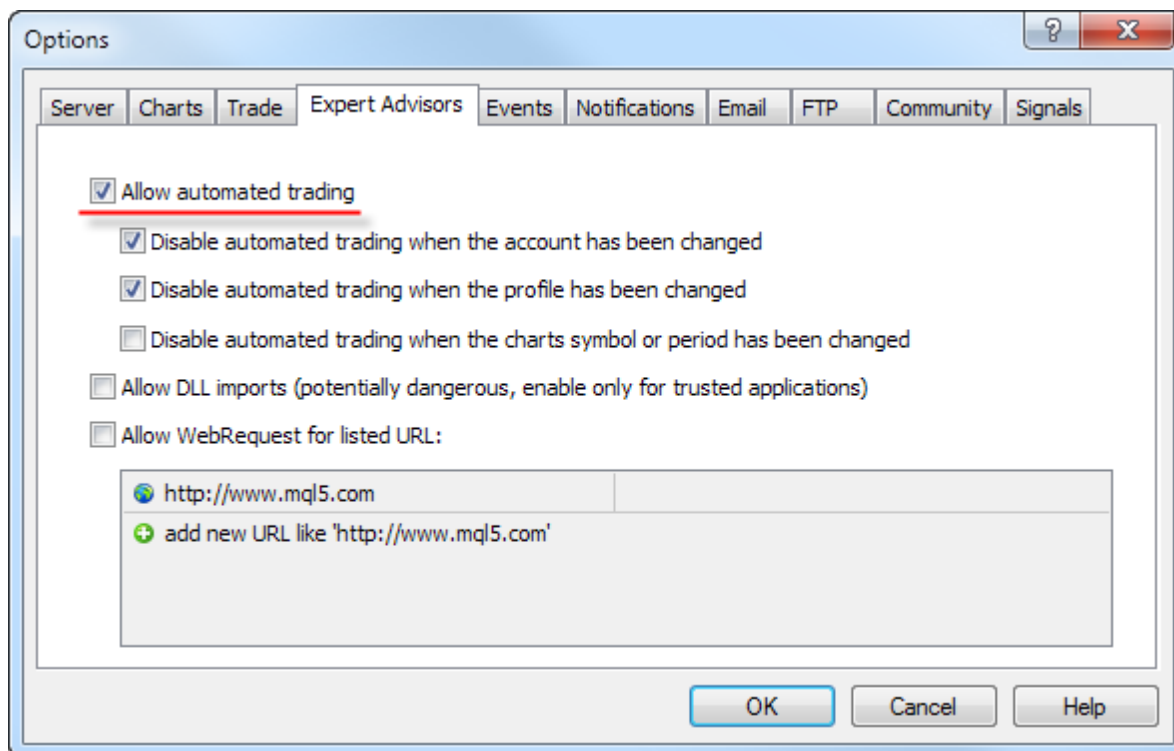
Handelsfunktionen können nur in der Experten und Skripte arbeiten, in Indikatoren ist Handel verboten.

### Überprüfung der Erlaubnis zum automatischen Handel

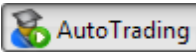
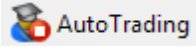
Um einen zuverlässigen Experten, der ohne menschliche Kontrolle automatisch funktionieren könnte, zu erstellen, ist es notwendig, eine Reihe von erforderlichen Kontrollen zu organisieren. Zuerst sollten wir programmatisch überprüfen, ob Handel überhaupt erlaubt ist. Dies ist eine Grundprüfung, die bei der Entwicklung eines automatisierten Systems unerlässlich ist.

### Überprüfung der Erlaubnis zum automatischen Handel im Terminal

Automatischer Handel für alle Programme kann im Terminal-Einstellungen verhindert oder erlaubt werden.



Sie können die Möglichkeit des automatisierten Handels direkt auf Standard-Panel des Terminals wechseln:

-  - automatischer Handel ist erlaubt, Verwendung von Handelsfunktionen in laufenden Anwendungen ist erlaubt.
-  - automatischer Handel ist verboten, die laufende Anwendungen funktionieren, aber die Handelsfunktionen können nicht ausgeführt werden.

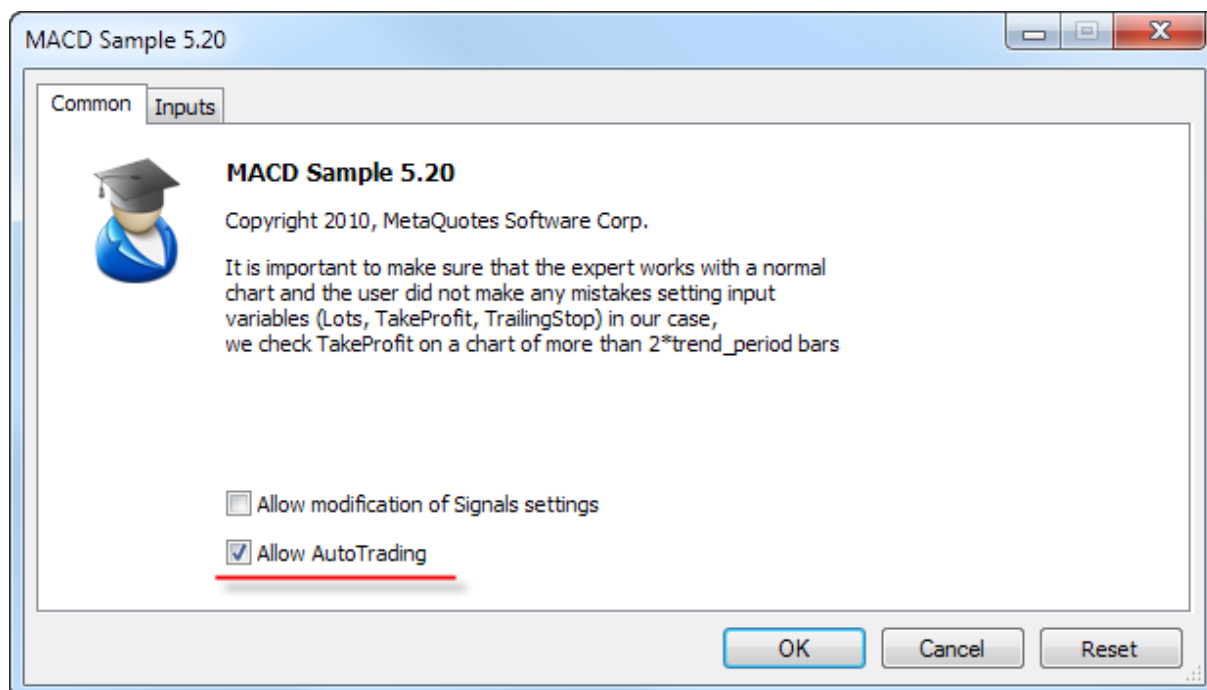
Beispiel der Überprüfung:

```
if (!TerminalInfoInteger(TERMINAL_TRADE_ALLOWED))
    Alert("Überprüfen Sie Erlaubnis zum automatischen Handel in Terminal-Einstellungen!");
```

## Überprüfung der Erlaubnis zum Handel für den laufenden Experte/Skript

Wenn Sie eine Anwendung starten, können Sie automatischen Handel speziell für sie erlauben oder verhindern. Um dies zu tun, verwenden Sie die spezielle Kontrollkästchen in den Programmeigenschaften.





Beispiel der Überprüfung:

```
if(!TerminalInfoInteger(TERMINAL_TRADE_ALLOWED))
    Alert("Überprüfen Sie Erlaubnis zum automatischen Handel in Terminal-Einstellungen")
else
{
    if(!MQLInfoInteger(MQL_TRADE_ALLOWED))
        Alert("Automatischer Handel ist verboten in Programmeigenschaften für ", __FILE__);
}
```

## Überprüfung der Erlaubnis zum Handel für Experten/Skripte für dieses Konto

Automatisierter Handel kann auf der Seite des Handelsservers verboten sein. Beispiel der Überprüfung:

```
if(!AccountInfoInteger(ACCOUNT_TRADE_EXPERT))
    Alert("Automatischer Handel ist für das Konto ", AccountInfoInteger(ACCOUNT_LOGIN),
        " auf der Seite des Handelsservers verboten");
```

Wenn automatisierter Handel ist für ein Handelskonto deaktiviert, werden Handelsoperationen von Experten/Skripte nicht ausgeführt.

## Überprüfung der Erlaubnis zum Handel für dieses Konto

In einigen Fällen sind alle Handelsoperationen für ein bestimmtes Handelskonto verboten - man kann nicht entweder manuell oder mit der Hilfe von Experten handeln. Beispiel der Überprüfung, wenn man zum Konto mit dem Investor-Passwort verbinden versuchte:

```
if(!AccountInfoInteger(ACCOUNT_TRADE_ALLOWED))
    Comment("Handel ist verboten für das Konto ", AccountInfoInteger(ACCOUNT_LOGIN),
        ".\n Vielleicht ist Investor-Passwort für die Verbindung mit dem Handelskonto\n\n Überprüfen Sie das Terminal-Protokoll, ob es einen solches Rekord gibt")
```

```
"\n\'", AccountInfoInteger(ACCOUNT_LOGIN), "\': trading has been disabled -
```

AccountInfoInteger(ACCOUNT\_TRADE\_ALLOWED) kann in folgenden Fälle false zurück geben:

- keine Verbindung mit dem Handelsserver. Kann durch TerminalInfoInteger(TERMINAL\_CONNECTED)) überprüft werden;
- Handelskonto ist auf den Modus read-only (in das Archiv gesendet) übertragen;
- Handel auf das Konto ist auf der Seite des Handelsservers verboten;
- Verbindung zum Handelskonto ist im Investor-Modus.

#### Sehen Sie auch

[Zustand des Terminals](#), [Information über das Konto](#), [Information über das ausgeführte MQL5-Programm](#)

## Ereignisse des Kundenterminals

### Init

Sofort nach Laden des Programms (des Experten oder des Benutzeranzeigers) und Start der Initialisierung der globalen Variablen wird das Ereignis Init gesendet, das durch die Funktion vom Kundenterminal [OnInit\(\)\\_verarbeitet](#) wird, wenn sie vorhanden ist. Dieses Ereignis wird auch nach Wechsel des Finanzinstrumentes und/oder der Chartperiode, nach Umcompilierung des Programms im Editor MetaEditor, nach Wechsel der Eingabeparameter vom Fenster der Einstellung von Expert oder Benutzerindikator. Ratgeber wird nach Kontowechsel initialisiert. Für Scripts wird das Ereignis Init nicht generiert.

### Deinit

Vor Deinitialisierung der globalen Variablen und Ausladen des Programms (des Experten oder des Benutzeranzeigers) sendet das Kundenterminal dem Programm das Ereignis Deinit. Das Ereignis Deinit wird auch nach Operationsende des Kundenterminals, beim Chartschliessen, unmittelbar vor dem Wechsel des Finanzinstrumentes und/oder der Chartperiode, bei erfolgreicher Umcompilierung des Programms, beim Wechsel der Eingabeparameter und beim Kontowechsel generiert.

[Grund der Deinitialisierung](#) kann man vom Parameter erhalten, der in die Funktion [OnDeinit\(\)](#) übertragen wird. Durchführen der Funktion OnDeinit() wird von 2.5 Sekunden begrenzt. Wenn die Funktion für diese Zeit ihre Arbeit nicht schliesst, wird ihre Ausführung zwangsläufig beendet. Für Scripts wird das Ereignis Deinit nicht generiert.

### Start

Ereignis [Start](#) - ist Sonderereignis für Scriptaktivierung nach seinem Laden. Dieses Ereignis wird durch die Funktion [OnStart](#) verarbeitet. Ereignis Start wird an Experten und Benutzeranzeiger nicht gesendet.

### NewTick

Ereignis NewTick wird beim Erscheinen der neuen Quotationen generiert und durch die Funktion [OnTick\(\)](#) der angehängten Ratgeber. Wenn beim Erscheinen der neuen Quotation die Funktion OnTick ausgeführt wurde, die in der früheren Quotation gestartet wurde, wird die erscheinende Quotation vom Ratgeber ignoriert, denn der entsprechende Wert wird nicht in die Queue der Expertereignisse eingereiht.

Alle während der Programmausführung erscheinenden Quotationen werden ignoriert bis Ausführung der Funktion OnTick() beendet. Dann wird die Funktion nach Erscheinen der neuen Quotation initialisiert. Das Ereignis NewTick wird, wenn Verwenden der Ratgeber (Schaltfläche "Ratgeber erlauben/verbieten") verboten ist.

Das Ereignis NewTick wird unabhängig davon generiert, ob der automatische Handel erlaubt oder verboten ist (Schaltfläche "Autohandel erlauben/verboten"). Verbot des automatischen Handels bedeutet nur Verbot, Handelsanforderungen aus dem Expert zu senden, die Arbeit des Experten wird nicht abgebrochen.

Verbot des automatischen Handels beim Drücken auf die angegebene Schaltfläche bricht nicht die laufende Ausführung der Funktion OnTick(). Funktion OnTick() wird nicht abgelaufen, wenn das Fenster der Funktionen des Ratgebers geöffnet wird.

## Calculate

Ereignis **Calculate** wird nur für Anzeiger sofort nach Senden vom Ereignis Init und bei jedem Wechsel der Preisdaten. Wird durch die Funktion [OnCalculate](#) verarbeitet.

## Timer

Ereignis **Timer** wird in regelmässige vom Kundenterminal für Experten generiert, der durch die Funktion generiert [EventSetTimer](#) Timer aktiviert hat. Gewöhnlich wird diese Funktion in der Funktion OnInit aufgerufen. Das Ereignis Timer wird durch Funktion [OnTimer](#) verarbeitet. Nach dem Operationsende des Experten muss man den erzeugten Timer durch die Funktion [EventKillTimer](#) entfernen, die gewöhnlich in der Funktion OnDeinit aufgerufen wird.

## Trade

Ereignis **Trade** wird beim Schliessen der Handelsoperation auf dem Handelsserver generiert. Verarbeitung des Ereignisses Trade wird durch die Funktion [OnTrade\(\)](#) für folgende Handelsoperationen durchgeführt:

- Einstellung, Modifikation oder Entfernung der wartenden Order;
- Abbruch der wartenden Order beim Geldmangel oder Gültigkeitsablauf;
- Aktivierung der wartenden Order ;
- Öffnen, Zusetzen oder Schliessen der Position (oder eines Teiles der Position);
- Modifikation der offenen Position (Veränderung von Stops).

## TradeTransaction

Als Ergebnis der Durchführung von bestimmten Aktionen auf dem Handelskonto, ändert sich der Kontostand. Zu solchen Aktionen gehören:

- Senden der Handelsanforderung aus einer beliebigen MQL5-Anwendung in dem Kundenterminal mittels der Funktionen [OrderSend](#) und [OrderSendAsync](#) und ihr nachfolgendes Ausführen;
- Senden der Handelsanforderung über die grafische Oberfläche des Terminals und ihr nachfolgendes Ausführen;
- Auslösen von aufgeschobenen Ordnern und Stop-Ordnern auf dem Server;
- Ausführen von Operationen auf der Seite der Handel-Server.

Als Ergebnis dieser Aktionen werden die folgenden Handelstransaktionen durchgeführt:

- Die Verarbeitung der Handelsanforderung;
- Änderung der offenen Aufträge;
- Änderung der Ordergeschichte;
- Änderung der Dealgeschichte;
- Änderung der Position.

Zum Beispiel, beim Senden Marktkauforder, wird Marktkauforder behandelt und eine entsprechende Marktkauforder für das Konto erstellt. Ist die Order durchgeführt, wird sie aus der Auftragsliste entfernt und zur Ordergeschichte hinzugefügt. Dann wird der entsprechende Deal zur Geschichte hinzugefügt und eine neue Position wird erstellt. Alle diese Aktionen sind Handelstransaktionen. Ankunft jeder solchen Transaktion im Terminal ist ein Ereignis TradeTransaction. Dieses Ereignis wird

durch die Funktion [OnTradeTransaction](#) verarbeitet.

### Tester

Ereignis [Tester](#) wird nach Testen des Experten in historischen Daten generiert. Bearbeitung des Ereignisses Tester wird durch die Funktion [OnTester\(\)](#) durchgeführt.

### TesterInit

Ereignis [TesterInit](#) wird beim Start der Optimierung im Strategie-Tester vor dem ersten Durchlauf generiert. Bearbeitung des Ereignisses TesterInit wird durch die Funktion [OnTesterInit\(\)](#) durchgeführt.

### TesterPass

Ereignis [TesterPass](#) wird beim Empfang eines neuen [Datenframes](#) generiert. Bearbeitung des Ereignisses TesterPass wird durch die Funktion [OnTesterPass\(\)](#) durchgeführt.

### TesterDeinit

Ereignis [TesterDeinit](#) wird nach der Optimierung des Expert Advisors im Strategie-Tester generiert. Bearbeitung des Ereignisses TesterDeinit wird durch die Funktion [OnTesterDeinit\(\)](#) durchgeführt.

### ChartEvent

Ereignis [ChartEvent](#) wird vom [Kundenterminal generiert](#) bei der Arbeit des Benutzers mit Chart:

- Keyboard Drücken, wenn Chartfenster fokussiert ist; ;
- Erzeugung des [graphischen Objekts](#);
- Entfernung des [graphischen Objekts](#);
- Mausklicken auf dem graphischen Objekt, das zum Chart gehört;
- Bewegung des graphischen Objekts mit der Maus;
- Ende des Texteditierens im Eingabefeld des graphischen Objektes `LabelEdit`.

Es gibt auch ein Benutzerereignis `ChartEvent`, das jedes mql5-Programm durch die Funktion [EventChartCustom](#) senden kann. Ereignis wird durch die Funktion [OnChartEvent](#) verarbeitet.

### BookEvent

Das Ereignis [BookEvent](#) wird vom Kundenterminal bei DOM Veränderung generiert und durch die Funktion [OnBookEvent](#) verarbeitet. Damit das Kundenterminal Ereignisse `BookEvent` für konkretes Symbol zu generieren anfängt, reicht es, auf Erhalten dieser Ereignisse durch die Funktion [MarketBookAdd](#) zu subscribieren.

Um die Subskription auf Erhalten des Ereignisses für konkretes Symbol zu annullieren, muss die Funktion `BookEvent` [MarketBookRelease](#) aufgerufen werden. Das Ereignis `BookEvent` ist broadcast - das bedeutet, wenn ein Expert auf Erhalten des Ereignisses `BookEvent` durch die Funktion `MarketBookAdd` subscribiert, werden alle anderen Experten mit dem Handler `OnBookEvent` dieses Ereignisses bekommen. Darum muss der Symbolname, der in Bearbeiter als Parameter übertragen wird, analysieren.

**Sehen Sie auch**

Funktionen der Ereignisverarbeitung, Programmausführung

## Ressourcen

### Benutzung der Grafik und Sound in MQL5 Programme

Die MQL5 Programme ermöglichen es Ihnen, mit Sound und Grafik-Dateien arbeiten:

- [PlaySound\(\)](#) spielt eine Sounddatei;
- [ObjectCreate\(\)](#) ermöglicht die Erstellung von Benutzeroberflächen mit Hilfe [grafischer Objekte](#) OBJ\_BITMAP und OBJ\_BITMAP\_LABEL .

### PlaySound()

Beispiel für Aufruf der Funktion [PlaySound\(\)](#):

```
//+-----+
//| Funktion ruft Standard-Funktion OrderSend() und spielt Sound |
//+-----+
void OrderSendWithAudio(MqlTradeRequest &request, MqlTradeResult &result)
{
    //--- Eine Anfrage an den Server senden
    OrderSend(request,result);
    //--- Wenn die Anfrage angenommen wird, Ok.wav spielen
    if(result.retcode==TRADE_RETCODE_PLACED) PlaySound("Ok.wav");
    //--- Wenn nicht, spielen Alarm aus Datei timeout.wav
    else PlaySound("timeout.wav");
};
```

Das Beispiel zeigt, wie Sounds aus Dateien Ok.wav und timeout.wav, die in das Standard-Terminal-Paket enthalten sind, spielen. Diese Dateien liegen im Ordner **Terminal\_Verzeichnis\Sounds**. Hier **Terminal\_Verzeichnis** bedeutet den Verzeichnis aus dem MetaTrader 5 Client Terminal gestartet wird. Der Standort des Terminals Verzeichnis kann direkt von einem MQL5 Programm in folgender Weise gefunden werden:

```
//--- Verzeichnis in dem Terminaldaten liegen
string terminal_path=TerminalInfoString(TERMINAL_PATH);
```

Sie können die Audio-Dateien nicht nur aus dem Ordner **Terminal\_Verzeichnis\Sounds** verwenden, sondern auch von jedem Unterordner im Ordner **Terminaldaten\_Verzeichnis\MQL5**. Lage des Verzeichnisses den Terminaldaten auf einem Computer können Sie durch das Terminal-Menü "File - Open Directory-Daten", oder programmatisch herausfinden:

```
//--- Verzeichnis in dem Terminaldaten liegen
string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
```

Zum Beispiel, wenn Demo.wav liegt in Terminaldaten\_Verzeichnis\MQL5\Files, dann Aufruf von PlaySound() soll auf folgende Weise geschrieben werden:

```
//--- Demo.wav aus Terminaldaten_Verzeichnis\MQL5\Files\Demo.wav spielen
PlaySound("\\Files\\Demo.wav");
```

Bitte beachten Sie, dass im Kommentar der Pfad zur Datei mit Backslash "\" geschrieben wird, und in

der Funktion "\\" verwendet wird.

Bei der Angabe des Pfades, immer nur den doppelten Backslash als Trennzeichen benutzen, da ein einzelner Backslash ist ein Kontrollsymbol des Compilers beim Umgang mit konstanten Zeichenfolgen und [Symbolkonstanten](#) im Programm-Quellcode.

Um Wiedergabe der Datei zu stoppen, rufen Sie die Funktion [PlaySound\(\)](#) mit Parameter NULL auf:

```
//--- Aufruf von Playsound() mit einem Parameter NULL stoppt Audiowiedergabe
PlaySound(NULL);
```

## ObjectCreate()

Ein Beispiel eines Expert Advisors, der mit Hilfe ObjectCreate() erzeugt ein Objekt "Grafischer Zeichen" (OBJ\_BITMAP\_LABEL).

```
string label_name="currency_label";           // Objektname OBJ_BITMAP_LABEL
string euro      ="\\Images\\euro.bmp";       // Pfad zu Terminaldaten_Verzeichnis\MQL5\T
string dollar    ="\\Images\\dollar.bmp";     // Pfad zu Terminaldaten_Verzeichnis\MQL5\T
//+-----+
//| Expert initialization function              |
//+-----+
int OnInit()
{
//--- Schaltfläche OBJ_BITMAP_LABEL erstellen, wenn es noch nicht erstellt worden
if(ObjectFind(0,label_name)<0)
{
//--- Versuchen wir OBJ_BITMAP_LABEL zu erstellen
bool created=ObjectCreate(0,label_name,OBJ_BITMAP_LABEL,0,0,0);
if(created)
{
//--- Binden wir die Schaltfläche zur linken oberen Ecke des Charts
ObjectSetInteger(0,label_name,OBJPROP_CORNER,CORNER_RIGHT_UPPER);
//--- Konfigurieren wir nun die Eigenschaften des Objekts
ObjectSetInteger(0,label_name,OBJPROP_XDISTANCE,100);
ObjectSetInteger(0,label_name,OBJPROP_YDISTANCE,50);
//--- Wird der letzte Fehlercode auf 0 fallen
ResetLastError();
//--- Laden wir das Bild für den Zustand der Schaltflächen "gedrückt"
bool set=ObjectSetString(0,label_name,OBJPROP_BMPFILE,0,euro);
//--- Ergebnis prüfen
if(!set)
{
PrintFormat("Bild konnte nicht aus der Datei %s geladen werden. Fehlercode
; }
ResetLastError();
//--- Laden wir das Bild für den Zustand der Schaltflächen "nicht gedrückt"
set=ObjectSetString(0,label_name,OBJPROP_BMPFILE,1,dollar);
```



```

        if(!set)
        {
            PrintFormat("Bild konnte nicht aus der Datei %s geladen werden. Fehlercode %d", label_name, GetLastError());
        }
        //---Geben wir dem Chart einen Befehl zu aktualisieren, so dass die Schaltfläche neu gezeichnet wird
        ChartRedraw(0);
    };
else
    {
        //--- Objekt konnte nicht erstellt werden, melden
        PrintFormat("Objekt OBJ_BITMAP_LABEL konnte nicht erstellt werden. Fehlercode %d", GetLastError());
    };
};
//---
return(INIT_SUCCEEDED);
};
//+-----+
//| Expert deinitialization function                                     |
//+-----+
void OnDeinit(const int reason)
{
    //--- Objekt aus dem Chart löschen
    ObjectDelete(0, label_name);
};

```

Erstellen und Konfigurieren eines grafischen Objekts namens `currency_label` sind in der Funktion `OnInit()` durchgeführt. Pfade zum grafischen Dateien werden in der [globalen Variablen](#) `euro` und `dollar` gegeben, doppelter Backslash für ein Separator verwendet ist:

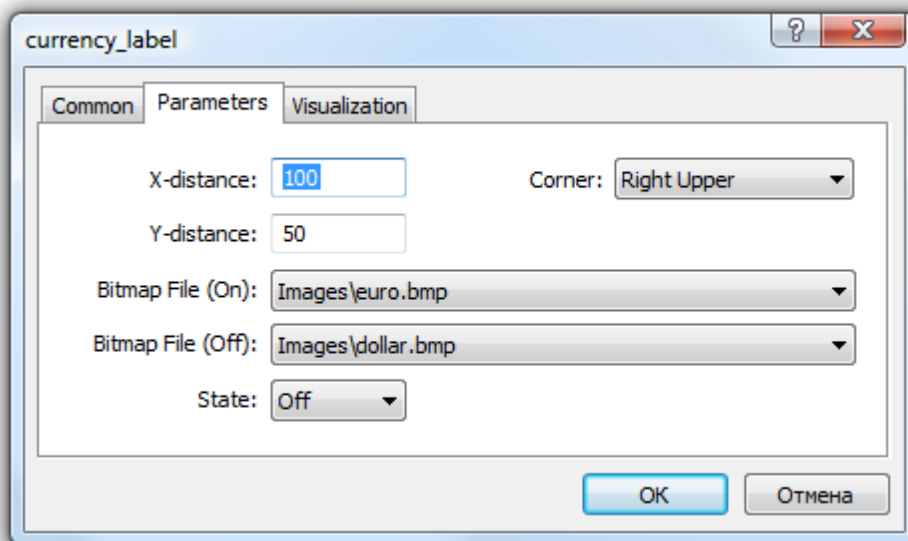
```

string euro      = "\\Images\\euro.bmp";    // Pfad zu Terminaldaten_Verzeichnis\MQL5\Images
string dollar    = "\\Images\\dollar.bmp";  // Pfad zu Terminaldaten_Verzeichnis\MQL5\Images

```

Die Dateien legen im Ordner `Terminaldaten_Verzeichnis\MQL5\Images`.

Objekt `OBJ_BITMAP_LABEL` ist eigentlich eine Schaltfläche, die eine der beiden Bilder zeigt, je nach Status der Schaltfläche (gedrückt oder nicht gedrückt): `euro.bmp` oder `dollar.bmp`.



Die Größe der Schaltfläche mit einer grafischen Oberfläche wird automatisch an die Größe des Bildes angepasst. Das Bild wird durch einen linken Mausklick auf das Objekt OBJ\_BITMAP\_LABEL geändert (im **Eigenschaften** soll "Disable selection" gesetzt werden). Objekt OBJ\_BITMAP wird die gleiche Weise erstellt - es ist für die Erstellung der Hintergrund mit einem notwendigen Bild verwendet.

Der Wert der [OBJPROP\\_BMPFILE](#) Eigenschaft, die verantwortlich für das Erscheinungsbild der Objekte OBJ\_BITMAP und OBJ\_BITMAP\_LABEL ist, kann dynamisch geändert werden. Dies ermöglicht das Erstellen verschiedener interaktiver Benutzeroberflächen für MQL5 Programme.

## Die Einschließung von Ressourcen in ausführbaren Dateien beim Kompilation der MQL5-Programme

Ein MQL5 Programm kann viele verschiedene herunterladbare Ressourcen in Form von Bild- und Audio-Dateien brauchen. Um die Notwendigkeit alle diese Dateien ins andere MQL5-Programm zu übertragen zu beseitigen, verwenden Sie die Compiler-Direktive **#resource**:

```
#resource Pfad_zu_Ressource-Datei
```

Befehl **#resource** sagt dem Compiler, dass die Ressource an dem angegebenen Pfad **Pfad\_zu\_Ressourcedatei** in die ausführbare Datei EX5 eingeschlossen werden sollten. Damit sind alle notwendigen Bilder und Sounds direkt in einer Datei EX5 gelegt, so dass es keine Notwendigkeit gibt, die Dateien zu übertragen, wenn Sie das Programm auf einem anderen Terminal ausführen möchten. Jede EX5 Datei kann Ressource haben, und jedes EX5 Programm kann Ressourcen aus einer anderen EX5 Programm verwenden.

BMP- und WAV-Dateien werden automatisch komprimiert, bevor sie in eine ausführbare Datei EX5 eingeschlossen sind. Dies bedeutet, dass die Verwendung der Ressourcen nicht nur erlaubt Ihnen MQL5 Programmen zu schaffen, sondern reduziert auch die Gesamtgröße der Dateien, die ein Chart für Grafiken und Sounds braucht.

Die Dateigröße der Ressource konnte nicht mehr als 16 MB sein.

## Suche nach bestimmten Ressourcen von einem Compiler

Eine Ressource ist mit dem Befehl `#resource "<Pfad zu Ressource Datei>"` eingefügt.

```
#resource "<Pfad_zu_Ressourcedatei>"
```

Die Länge des konstanten String `<Pfad_zu_Ressourcedatei>` sollte nicht mehr als 63 Zeichen.

Der Compiler sucht nach einer Ressource auf dem angegebenen Pfad in folgender Reihenfolge:

- wenn der Backslash "\" Separator (geschrieben "\\") wird am Anfang des Pfads platziert, sucht es für die Ressource relativ zum Verzeichnis **Terminaldaten\_Verzeichnis\MQL5**,
- wenn es kein Backslash gibt, wird die Ressource auf den Speicherort der Quelldatei, in der diese Ressource ist vorgeschrieben, gesucht.

Die Ressource-Pfad kann man nicht die Teilstrings `"..\\"` und `"\\"` verwenden.

Beispiele:

```
//--- Korrekte Angabe von Ressourcen
#resource "\\Images\\euro.bmp" // euro.bmp befindet sich in Terminaldaten_Verzeichnis\
#resource "picture.bmp"        // picture.bmp befindet sich in dasselbe Verzeichnis, v
#resource "Resource\\map.bmp"  // Ressource liegt in Quelldatei_Verzeichnis\Resource\

//--- Falsche Angabe von Ressourcen
#resource ":picture_2.bmp"      // ":" kann nicht verwendet werden
#resource "..\\picture_3.bmp"   // ".." kann nicht verwendet werden
#resource "\\Files\\Images\\Folder_First\\My_panel\\Labels\\too_long_path.bmp" //mehr
```

## Die Verwendung der Ressourcen

### Ressource-Name

Nachdem die Ressource mit der Direktive `#resource` deklariert ist, kann es in jedem Teil des Programms verwendet werden. Ressource-Name wird seinen Pfad ohne einen Backslash am Anfang des Strings, die den Pfad zu der Ressource definiert. Um eine eigene Ressource zu verwenden, spezielles Attribut `::` soll vor dem Ressource-Namen hinzugefügt werden.

Beispiele:

```
//--- Beispiele der Ressourcen und ihre Namen in den Kommentaren
#resource "\\Images\\euro.bmp"           // Resource-Name - Images\euro.bmp
#resource "picture.bmp"                  // Resource-Name - picture.bmp
#resource "Resource\\map.bmp"            // Resource-Name - Resource\map.bmp
#resource "\\Files\\Pictures\\good.bmp"  // Resource-Name - Files\Pictures\good.bmp
#resource "\\Files\\Demo.wav";           // Resource-Name - Files\Demo.wav"
#resource "\\Sounds\\thrill.wav";        // Resource-Name - Sounds\thrill.wav"
...

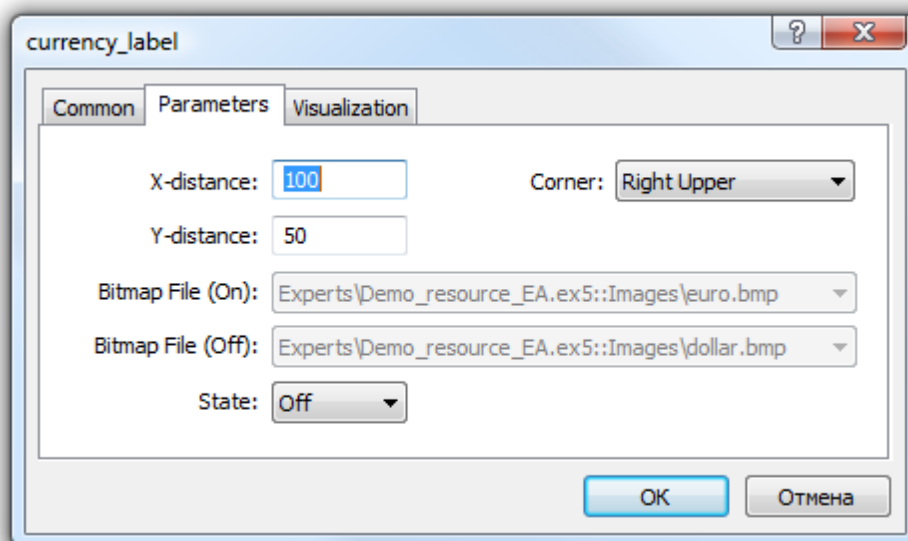
```

```
//--- Verwendung der Ressourcen
ObjectSetString(0,bitmap_name,OBJPROP_BMPFILE,0,"::Images\\euro.bmp");
...
ObjectSetString(0,my_bitmap,OBJPROP_BMPFILE,0,"::picture.bmp");
...
set=ObjectSetString(0,bitmap_label,OBJPROP_BMPFILE,1,"::Files\\Pictures\\good.bmp");
...
PlaySound("::Files\\Demo.wav");
...
PlaySound("::Sounds\\thrill.wav");
```

Anzumerken ist, dass bei der Einstellung von Bildern aus einer Ressource an den OBJ\_BITMAP und OBJ\_BITMAP\_LABEL Objekten, der Wert der Eigenschaft OBJPROP\_BMPFILE kann nicht manuell geändert werden. Zum Beispiel für das Erstellen OBJ\_BITMAP\_LABEL, verwenden wir unserer Ressourcen euro.bmp und dollar.bmp.

```
#resource "\\Images\\euro.bmp"; // euro.bmp liegt in Terminaldaten_Verzeichnis\MQL5
#resource "\\Images\\dollar.bmp"; // dollar.bmp liegt in Terminaldaten_Verzeichnis\MQ
```

Dann, in der Eigenschaften dieses Objekts sehen wir, dass die Eigenschaften BitMap File (On) und BitMap File (Off) sind grau und können nicht manuell geändert werden:



## Verwendung der Ressourcen anderer mql5-Programmen

Die Verwendung der Mittel hat einen weiteren Vorteil - in jedem MQL5-Programm kann Ressource aus einer beliebigen Datei EX5 verwendet werden. So können Ressourcen aus einer Datei EX5 in vielen anderen MQL5-Programmen verwendet werden.

Um eine Ressource-Name aus einer anderen Datei zu verwenden, soll sie als <Pfad\_EX5\_Dateiname> angegeben werden. Zum Beispiel, in Skript Draw\_Triangles\_Script.mq5 wird eine Ressource zum Bild in der Datei triangle.bmp angegeben:

```
#resource "\\Files\\triangle.bmp"
```

Dann ist ihr Name "Files\triangle.bmp" für das Skript. Um es zu verwenden, soll Attribut "::" zum Ressource-Namen hinzugefügt werden.

```
//--- Verwendung der Ressource in dem Skript
ObjectSetString(0,my_bitmap_name,OBJPROP_BMPFILE,0,"::Files\\triangle.bmp");
```

Um die gleiche Ressource aus einem anderen Programm zu verwenden, z. B. aus einem Expert Advisor, müssen wir auf die Ressource Name fügen den Pfad zu der EX5-Datei relativ zu Terminaldaten\_Verzeichnis\MQL5\ und den Namen der EX5-Datei des Skripts - Draw\_Triangles\_Script.ex5 hinzufügen. Nehmen wir an, dass das Skript liegt in der Standard-Ordner Terminaldaten\_Verzeichnis\MQL5\Scripts\, dann soll der Anruf wie folgt geschrieben werden:

```
//--- Verwendung der Ressource aus dem Skript im Expert Advisor
ObjectSetString(0,my_bitmap_name,OBJPROP_BMPFILE,0,"\\Scripts\\Draw_Triangles_Script.e
```

Wenn beim Zugriff auf Ressourcen in einer anderen EX5-Datei gibt es kein Pfad zur ausführbaren Datei, dann ist die ausführbare Datei im selben Ordner, wo das Programm das ruft die Ressource liegt, gesucht. Das heißt, wenn der Expert Advisor fordert eine Ressource aus Datei Draw\_Triangles\_Script.ex5 ohne Angabe des Pfades, etwa so:

```
//--- Aufruf der Ressource des Skripts im Expert Advisor ohne Pfad
ObjectSetString(0,my_bitmap_name,OBJPROP_BMPFILE,0,"Draw_Triangles_Script.ex5::Files\\
```

dann wird die Datei in Terminaldaten\_Verzeichnis\MQL5\Experts\ durchgesucht, wenn der Expert Advisor sich in Terminaldaten\_Verzeichnis\MQL5\Experts\ befindet.

## Arbeiten mit als Ressourcen verbundenen benutzerdefinierten Indikatoren

MQL5-Programme können eine oder mehrere benutzerdefinierte Indikatoren brauchen, können sie alle in den Code der ausführbaren MQL5-Programm aufgenommen werden. Aufnahme von Indikatoren als Ressourcen vereinfacht die Verteilung von Programmen.

Beispiel für den Anschluss und Verwendung eines benutzerdefinierten Indikators SampleIndicator.ex5 im Verzeichnis: terminal\_data\_folder\MQL5\Indicators\:

```
//+-----+
//|                                     SampleEA.mq5 |
//|                                     Copyright 2013, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#resource "\\Indicators\\SampleIndicator.ex5"
int handle_ind;
//+-----+
//| Expert initialization function |
//+-----+
int OnInit()
{
//---
    handle_ind=iCustom(_Symbol,_Period,"::Indicators\\SampleIndicator.ex5");
    if(handle_ind==INVALID_HANDLE)
```

```

    {
        Print("Expert: iCustom call: Error code=", GetLastError());
        return(INIT_FAILED);
    }
    //--- ...
    return(INIT_SUCCEEDED);
};

```

Der Fall, wo der benutzerdefinierte Indikator in der Funktion [OnInit\(\)](#) erzeugt eine oder mehrere Kopien von sich selbst, erfordert gesonderte Betrachtung. Für die Nutzung der Ressource aus einer MQL5-Programm ist es notwendig, die Ressource in der folgenden Form anzugeben: <Pfad\_Dateiname\_EX5>::<Ressourcename>.

Zum Beispiel, wenn der Indikator SampleIndicator.ex5 ist in Expert Advisor SampleEA.ex5 als Ressource aufgenommen ist, wird der Pfad zu sich selbst, der bei der Aufruf von [iCustom\(\)](#) in der Funktion der Initialisierung der benutzerdefinierten Indikator angegeben war, wird wie folgt sein: "\\Experts\\SampleEA.ex5::Indicators\\SampleIndicator.ex5". Wenn Sie diesen Pfad explizit angeben, ist der benutzerdefinierte Indikator SampleIndicator.ex5 fest mit dem Expert Advisor SampleEA.ex5 gebunden und verliert die Fähigkeit selbst zu arbeiten.

Ein Pfad zu sich selbst kann mit der Funktion [GetRelativeProgramPath\(\)](#) erhalten werden. Hier ist ein Beispiel davon:

```

//+-----+
//|                                     SampleIndicator.mq5 |
//|                                     Copyright 2013, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property indicator_separate_window
#property indicator_plots 0
int handle;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- Der falsche Weg der Angabe einer Referenz auf sich selbst
    //--- string path="\\Experts\\SampleEA.ex5::Indicators\\SampleIndicator.ex5";
    //--- Der richtige Weg vom Erhalten einer Referenz auf sich selbst
    string path=GetRelativeProgramPath();
    //--- indicator buffers mapping
    handle=iCustom(_Symbol,_Period,path,0,0);
    if(handle==INVALID_HANDLE)
    {
        Print("Indicator: iCustom call: Error code=", GetLastError());
        return(INIT_FAILED);
    }
    else Print("Indicator handle=", handle);
    //---
    return(INIT_SUCCEEDED);
}

```

```

; }
///....
//+-----+
//| GetRelativeProgramPath |
//+-----+
string GetRelativeProgramPath()
{
    int pos2;
    ///--- erhalten wir den absoluten Pfad zum Programm
    string path=MQLInfoString(MQL_PROGRAM_PATH);
    ///--- Finden Sie die Position eines Teilstrings "\\MQL5\\"
    int pos =StringFind(path,"\\MQL5\\");
    ///--- Teilstring ist nicht gefunden - Fehler
    if(pos<0)
        return(NULL);
    ///--- Verzeichnis "\\MQL5" auslassen
    pos+=5;
    ///--- extra '\\' auslassen
    while(StringGetCharacter(path,pos+1)=='\\')
        pos++;
    ///--- Wenn es eine Ressource ist, geben wir den Pfad relativ zu MQL5-Verzeichnis zurück
    if(StringFind(path,":",pos)>=0)
        return(StringSubstr(path,pos));
    ///--- Finden wir ein Trennzeichen für die erste Unterverzeichnis in MQL5 (zB MQL5\Indi
    ///--- Wenn es gibt es nicht, geben wir den Pfad relativ zu MQL5-Verzeichnis zurück
    if((pos2=StringFind(path,"\\",pos+1))<0)
        return(StringSubstr(path,pos));
    ///--- geben wir den Pfad relativ zu Verzeichnis zurück (zB MQL5\Indicators)
    return(StringSubstr(path,pos2+1));
; }
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const int begin,
               const double& price[])
{
    ///--- return value of prev_calculated for next call
    return(rates_total);
}

```

#### Sehen Sie auch

[ResourceCreate\(\)](#), [ResourceSave\(\)](#), [PlaySound\(\)](#), [ObjectSetInteger\(\)](#), [ChartApplyTemplate\(\)](#),  
[Dateioperationen](#)

## Aufruf der Importfunktionen

für Import der Funktionen während der Ausführung des mql5-Programmes wird frühe Bindung verwendet. Das bedeutet wenn es im Programm Aufruf der Importfunktion gibt, wird der entsprechende Modul (ex5 oder dll) beim Programmladen geladen. Bibliotheken MQL5 und DLL Werden im Thread des Aufrufmoduls durchgeführt.

Es ist nicht empfehlenswert, den vollstaendig bestimmten Namen des geladenen Moduls der Art *Drive: \Directory\FileName.Ext* verwenden. Bibliotheken MQL5 werden vom Ordner *terminal\_dir\MQL5\Libraries* ausgeladen. Wenn Bibliothek nicht gefunden wurde, versucht man, Bibliothek vom Ordner *terminal\_dir\experts* herunterzuladen.

Systembibliotheken werden nach Regeln (DLL) des Operationssystems geladen. Wenn die Bibliothek schon geladen wird (zB. von anderen Experten oder aus anderem Kundenterminal, das parallel startet) wird die schon geladene Bibliothek aufgerufen. Anderenfalls erfolgt die Suche in der Folge:

1. Verzeichnis, von dem das dll importierende Modul abgelaufen wurde. Unter Modul wird Expert, Script, Indikator oder Bibliothek EX5 verstanden;
2. Verzeichnis *Katalog\_Data\_Terminals\MQL5\Libraries* ([TERMINAL\\_DATA\\_PATH\MQL5\Libraries](#));
3. Verzeichnis, von dem das Kundenterminal MetaTrader 5 gestartet wurde;
4. Systemverzeichnis;
5. Verzeichnis Windows;
6. Laufendes Verzeichnis;
7. Verzeichnisse, die in Systemvariable PATH aufgelistet sind.

Wenn die Bibliothek DLL eine andere DLL in ihrer Arbeit verwendet, dann kann die erste DLL beim Fehlen der zweiten DLL nicht geladen werden.

Vor dem Laden des Experten (Script, Indikator) wird die gesamte Liste aller Bibliothekmoduln EX5 gebildet, die sowohl von dem geladenen Experten (Script, Indikator), als auch von der Bibliothek dieser Liste zu verwenden gedacht. So erfolgt das einmalige Laden der mehrfach verwendeten Bibliothekmoduln EX5. Bibliotheken verwenden [vorbestimmte Variablen](#) des Experten (Script, Indikator), der sie aufgerufen hat.

Suche der importierten Bibliothek wird in der Folge durchgeführt:

1. Verzeichnis, Pfad zu dem in Bezug auf Verzeichnis des importierenden EX5 Experten (Script, Indikator) vorgegeben wird;
2. Verzeichnis *Katalog\_Terminals\MQL5\Libraries*;
3. Verzeichnis *MQL5\Libraries* im Gesamtverzeichnis aller Kundenterminals MetaTrader 5 (*Common\MQL5\Libraries*).

Funktionen, die aus DLL in mql5-Programm , [importiert werden](#), müssen Windows API Abkommen garantieren. Für Garantieren dieses Abkommens wird im Ausgangstext der Programme, die mit den Sprachen C oder C++ geschrieben werden, das Schluesselwort `__stdcall` verwendet, das für Compiler der Gesellschaft Microsoft(r) spezifisch ist. Das Abkommen ist dadurch charakterisiert:

- ausrufende Funktion (hier mql5-Programm) muss "sehen" Prototyp der aufgerufenen Funktion (importierten aus DLL), um Parameter zum Stack richtig zusammenzustellen;
- ausrufende Funktion (hier mql5-Programm) stellt Parameter auf Stack in umgekehrter Reihenfolge, von rechts nach links - gerade in dieser Reihenfolge liest die importierte Funktion die ihr



übertragenen Parameter ein;

- Parameter werden nach Wert übertragen ausser denen, die explizit durch Referenz übertragen werden (in diesem Fall sind es Zeilen)
- Importfunktion, klärt Stack selbst ab beim Einlesen der ihr übertragenen Parameter.

Beim Beschreiben des Prototyps der Importfunktion können Parameter mit Defaultwerten verwendet werden.

Falls die entsprechende Bibliothek nicht in der Lage war, geladen zu werden oder es ist untersagt, DLL zu verwenden oder die importierte Funktion nicht gefunden war - stoppt Expert seine Arbeit mit der entsprechenden Nachricht "expert stopped" im Journal. Dabei wird Expert nicht ablaufen lassen, bevor er neuinitialisiert wird. Expert kann neuinitialisiert werden im Ergebnis der Umcompilierung oder der Öffnung der Eigenschaften der Tabelle und Drücken der Schaltfläche OK.

## Parameterübertragung

Alle Parameter der [einfachen Typen](#) werden nach Wert übertragen, wenn nicht explizit angegeben wird, dass sie durch Referenz übertragen werden. Bei der Übertragung einer [Zeile](#) wird Pufferadresse der kopierten Zeile übertragen; wenn eine Zeile durch Referenz übertragen wird, wird in die aus DLL importierte Funktion Pufferadresse gerader dieser Zeile Zeile ohne Kopieren übertragen.

[Strukturen](#), die dynamische Felder, Zeilen, Klassen, andere komplizierte Strukturen, aber auch statische oder [dynamische Felder](#) der aufgelisteten Objekte enthalten, können nicht als Parameter in eine importierte Funktion übertragen werden.

Bei der Übertragung des Feldes in DLL wird immer (unabhängig von der Flagge [AS\\_SERIES](#)) die Adresse des Pufferanfangs übertragen. Funktion innerhalb DLL weiss nichts von der Flagge AS\_SERIES, das übertragene Feld ist ein statisches Feld der unbekannten Länge, für Spezifizieren der Feldgrösse verwenden Sie einen zusätzlichen Parameter.

## Ausführungsfehler

Im ausführenden Subsystem des Kundenterminals kann man den [Fehlerkode](#) aufbewahren, wenn er bei der Ausführung des mql5-Programms entsteht. für jedes ausführbare mql5-Programm gibt es die vorbestimmte Variable [\\_LastError](#).

Vor dem Start der Funktion [OnInit](#) wird die Variable `_LastError` ausgenullt. Bei der fehlerhaften Situation während der Berechnungen oder beim Aufruf der eingebauten Funktion akzeptiert die Variable `_LastError` Kode des entsprechenden Fehlers. Den Wert, der in dieser Variable aufbewahren wird, kann man durch die Funktion [GetLastError\(\)](#) erhalten.

Es gibt eine Reihe kritischer Fehler, deren Entstehung das Programm sofort brechen:

- Dividieren durch Null;
- Feldbereichsüberschreitung;
- Verwendung des unkorrekten [Objektanzeigers](#).

## Testen von Handelsstrategien

Die Idee des automatisierten Handels ist attraktiv, weil ein Handelsroboter unermüdlich 24/7 arbeiten kann. Der Roboter wird nicht müde, zweifelhaft und hat keine Angst, er ist völlig frei von irgendwelchen psychischen Problemen. Sie müssen nur klar die Handelsregeln formalisieren und sie in den Algorithmen implementieren, und der Roboter ist fertig zu arbeiten. Aber zuerst müssen Sie sicherstellen, dass die folgenden zwei wichtige Bedingungen erfüllt sind:

- Der Expert Advisor führt [Handelsoperationen](#) in Übereinstimmung mit den Regeln des Handelssystems;
- Die Handelsstrategie, die im EA implementiert ist, zeigt einen Gewinn aus der Geschichte.

Um Antworten auf diese Fragen zu bekommen, verwenden wir den [Strategie-Tester](#), der ein Teil von MetaTrader 5 Client Terminal ist.

In diesem Abschnitt werden die Merkmale von Testen und Optimierung der Programme in der Strategie-Tester beschrieben:

- [Beschränkungen für die Funktionen im Strategie-Tester](#)
- [Modi Von Tick-Generierung](#)
- [Simulation von Spread](#)
- [Globale Variablen des Kundenterminals](#)
- [Berechnung von Indikatoren beim Testen](#)
- [Herunterladen von Geschichte beim Testen](#)
- [Multi-Currency-Testen](#)
- [Simulation der Zeit im Tester](#)
- [Grafische Objekte beim Testen](#)
- [OnTimer\(\)-Funktion im Tester](#)
- [Sleep\(\)-Funktion im Tester](#)
- [Print\(\)-Funktion im Tester](#)
- [Verwenden des Testers für Optimierungsprobleme in mathematischen Berechnungen](#)
- [Synchronization von Balken beim Testen in "Open Prices Only"-Modus](#)
- [IndicatorRelease\(\)-Funktion im Tester](#)
- [Behandlung von Ereignissen im Tester](#)
- [Testagenten](#)
- [Der Datenaustausch zwischen dem Terminal und dem Agenten](#)
- [Verwendung von einen freigegebenen Ordner von allen Kundenterminals](#)
- [Verwendung von DLL](#)

## Beschränkungen für die Funktionen im Strategie-Tester

Es gibt Beschränkungen für einige Funktionen im Strategie-Tester von Client Terminal.

## Funktionen Print() und PrintFormat()

Um die Leistung während Optimierung der Parameter von Experten zu erhöhen, werden die Funktionen [Print\(\)](#) und [PrintFormat\(\)](#) nicht ausgeführt. Die Ausnahme ist die Verwendung dieser Funktionen innerhalb des [OnInit\(\)](#)-Handlers. Dies erleichtert es, die Ursachen von Fehlern zu suchen, wenn sie auftreten.

## Funktionen Alert(), MessageBox(), PlaySound(), SendFTP, SendMail(), SendNotification(), WebRequest()

Die Funktionen der Interaktion mit der "Außenwelt" [Alert\(\)](#), [MessageBox\(\)](#), [PlaySound\(\)](#), [SendFTP\(\)](#), [SendMail\(\)](#), [SendNotification\(\)](#) und [WebRequest\(\)](#) werden im Strategie-Tester nicht ausgeführt.

## Modi Von Tick-Generierung

Ein Expert Advisor auf MQL5 ist ein Programm, das jedes Mal als Reaktion auf eine äußere Einwirkung - [Ereignis](#) - läuft. Für jedes vordefinierte Ereignis hat der EA die Funktion, die diesem Ereignis entspricht - [Event-Handler](#).

Das wichtigste Ereignis für den Expert Advisor ist eine Preisänderung - [NewTick](#), und deshalb, um Expert Advisors zu testen, müssen wir Tick-Sequenz generieren. Es gibt drei Modi von Tick-Generation im Strategie-Tester von MetaTrader 5 Kundenterminal:

- Jeder Tick
- 1 Minute OHLC (OHLC Preise mit 1 Minute-Balken)
- Nur Eröffnungspreise

Die Grund- und detaillierteste ist der "Jeder Tick"-Modus. Sind die zwei anderen Modi die Vereinfachungen des basis Modus "Jeder Tick". Betrachten wir alle drei Modi, um die Unterschiede zwischen ihnen zu verstehen.

### Jeder Tick

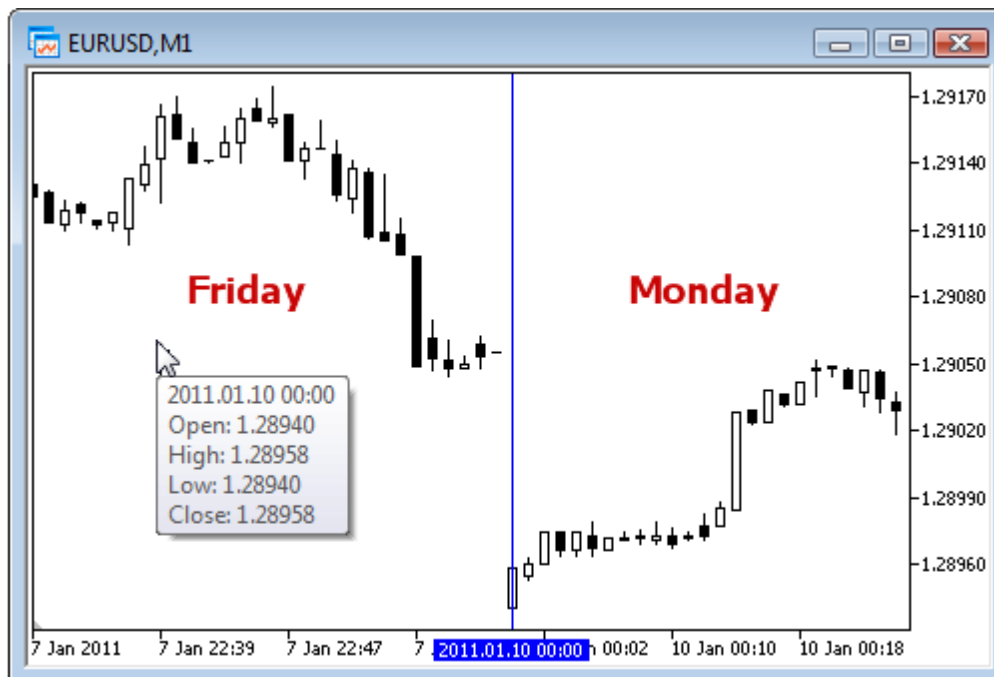
Die historische Preisdaten von Finanzinstrumenten werden aus dem Handelsserver an den MetaTrader 5 Kundenterminal in Form von sparsam verpackten Blöcke von 1-Minuten Balken übertragen. Für weitere Informationen über die Abfrage und Bauen der gewünschten Zeitrahmen lesen Sie bitte die Hilfethema [Datenzugriff organisieren](#).

Das minimale Element der Preisgeschichte ist 1-Minute Balken, von dem Sie Informationen über die vier Werte der Preise bekommen können:

- Open - Preis, zu dem der 1-Minute Balken eröffnet wurde;
- High - das Maximum, das während dieses 1-Minute Balkens erreicht wurde;
- Low - das Minimum, das während dieses 1-Minute Balkens erreicht wurde;
- Close - Schlusspreis des Balkens.

Ein neue 1-Minute-Balken öffnet nicht beim Start einer neuen Minute (Anzahl der Sekunden ist gleich 0), aber wenn ein Tick kommt - eine Preisänderung mindestens auf einen Punkt. Die Abbildung zeigt den ersten 1-Minute-Balken des neuen Handelswoche, der die Öffnungszeit des 2011.01.10 00:00 Uhr

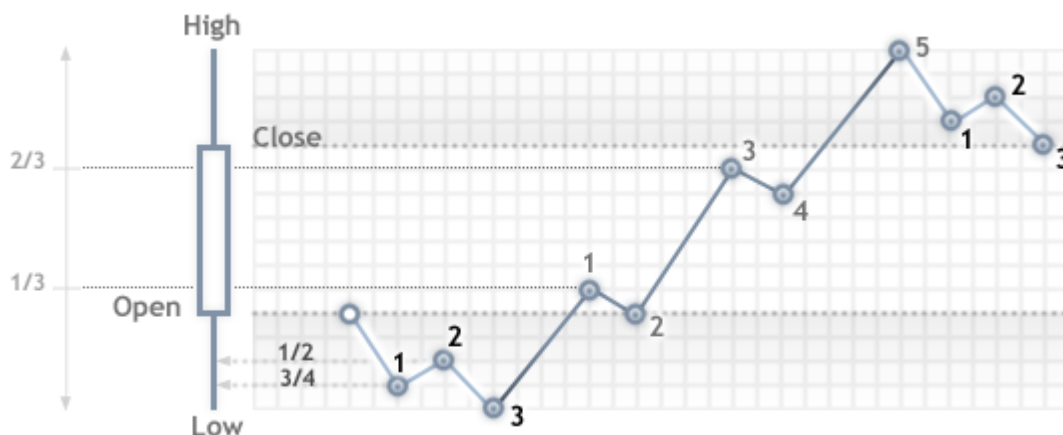
hat. Der Preisunterschied zwischen Freitag und Montag, den wir auf dem Chart sehen, ist weit verbreitet, da Wechselkurse schwankt auch am Wochenende als Reaktion auf eingehende Nachrichten.



Für diesen Balken wir wissen nur, dass der 1-Minute-Balken auf 10 Januar 2011 in 00 Stunden 00 Minuten geöffnet wurde, aber wir wissen nichts über die Sekunden. Er könnte am 00.00.12 oder 00.00.36 (12 oder 36 Sekunden nach dem Start eines neuen Tages) oder jedem anderen Zeitpunkt innerhalb dieser Minute geöffnet worden sein. Aber wir wissen, dass der Open-Preis von EURUSD war 1,28940 bei der Eröffnung des neuen 1-Minute-Balkens.

Ebenso wissen wir nicht die Sekunde, wenn der Tick kam, der den entsprechenden Schlusskurs dieses 1-Minute-Balkens entspricht. Wir wissen nur das es der letzte Preis auf diesem 1-Minute-Balken ist, der als Close-Preis aufgezeichnet wurde. Für diesen Minuten war der Preis 1,28958. Die Zeit des Auftretens von High- und Low-Preisen ist ebenfalls unbekannt, aber wir wissen, dass die maximalen und minimalen Preisen die Ebenen von 1,28958 und 1,28940 beziehungsweise besucht haben.

Um die Handelsstrategie zu testen, brauchen wir eine Folge von Ticks, auf der die Arbeit des Expert Advisors simuliert wird. Damit für jeden 1-Minute-Balken wissen wir die vier **Kontrollpunkte**, wo der Preis auf jeden Fall gewesen ist. Wenn der Balken nur 4 Ticks hat, dann sind diese Informationen ausreichend für Testen, aber in der Regel ist Tick-Volumen mehr als 4. Daher ist es notwendig, zusätzliche Kontrollpunkte für Ticks zu generieren, die zwischen den Preisen Open, High, Low und Close kam. Das Prinzip der Generierung von Ticks in der "Jeder Tick"-Modus wird in dem Artikel [The Algorithm of Ticks' Generation within the Strategy Tester of the MetaTrader 5 Terminal](#), beschrieben, eine Abbildung von denen unten gezeigt ist.



Beim Testen im "Jeder Tick"-Modus, wird die Funktion [OnTick\(\)](#) von EA an jedem Kontrollpunkt aufgerufen. Jeder Kontrollpunkt ist ein Tick von generierten Sequenz. Der Expert Advisor erhält die Zeit und den Preis des simulierten Ticks, so wie es wäre, wenn Sie online arbeiten.

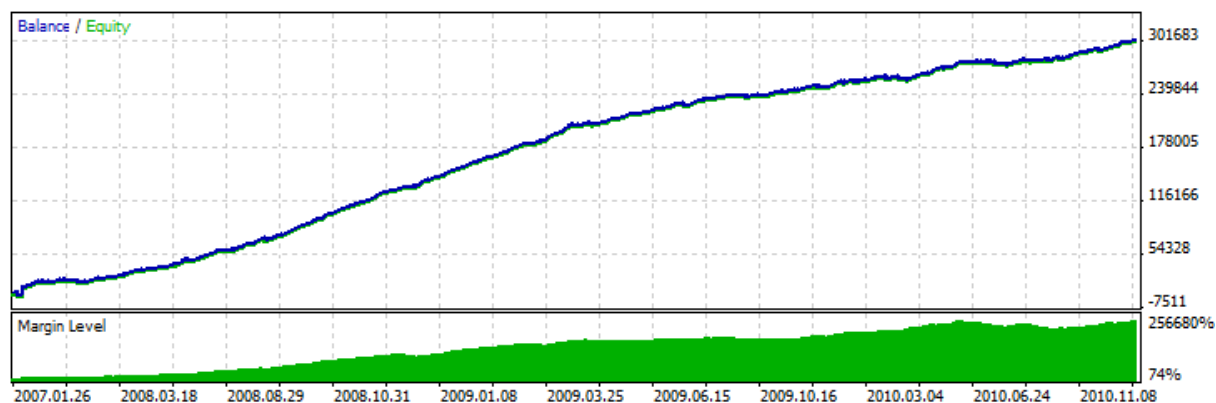
**Wichtig:** "Jeder Tick" ist der genauesten Test- Modus, aber zur gleichen Zeit nimmt er am meisten Zeit in Anspruch. Für die erste Auswertung der meisten Handelsstrategien ist in der Regel ausreichend, eine der beiden anderen Test-Modi zu verwenden.

## 1 minute OHLC

"Jeder Tick" ist ist der genaueste der drei Modi, aber zur gleichen Zeit, ist er der langsamste. Der Lauf der OnTick()-Handler erfolgt bei jedem Tick, während Tick-Volumen ziemlich groß sein können. Für eine Strategie, bei der die Ticksfolge von Kursbewegungen während des Balkens keine Rolle spielt, gibt es ein schneller und rauer Simulationsmodus - "1 Minute OHLC".

Im "1 minute OHLC"-Modus wird Tick-Sequenz nur auf der Basis von **OHLC-Preisen von 1-Minute-Balken** gebaut, wird die Anzahl der generierten Kontrollpunkte deutlich reduziert - damit die Testzeit auch reduziert. Der Start von OnTick()-Funktion wird auf alle Kontrollpunkte gemacht, die auf die Preise OHLC von 1-Minute\_Balken gebaut werden.

Die Weigerung, zusätzliche Ticks zwischen den Preisen von Open, High, Low und Close zu generieren, führt zu einer starren Determinismus in der Entwicklung des Preises vom Moment, wenn der Open-Preis bestimmt wird. Dies macht es möglich, einen "Gral des Tests" zu erstellen, der einen schönen aufsteigenden Bilanz-Chart für Tests zeigt. Ein Beispiel von solchem Gral ist in Code Base verfügbar - [Grr-al](#).



Die Abbildung zeigt einen sehr attraktiven Chart von Testen dieses Expert Advisors. Wie wird es aufgenommen? Wir wissen 4 Preise für den 1-Minute-Balken, und wir wissen auch, dass der erste Preis ist Open, und der letzte ist Close. Zwischen ihnen sind High und Low, ist die Reihenfolge ihres Auftretens nicht bekannt, aber es ist bekannt, dass der Preis High größer als oder gleich zu Open ist (Low ist kleiner oder gleich dem Preis Open).

Ist ausreichend genug, die Zeit von Erhalt des Open-Preises zu ermitteln, und dann den nächsten Tick zu analysieren, um festzustellen, ob es High oder Low ist. Wenn der Preis liegt unter dem Open-Preis, dann haben wir den Low-Preis - kaufen wir auf diesem Tick, wird der nächste Tick dem High-Preis entsprechen, an denen wir schließen Buy und öffnen Sell. Der nächste Tick ist der letzte, er ist der Close-Preis, und wir schließen den Verkauf (Sell) auf ihn.

Wenn nach dem Preis wir einen Tick mit einem Preis größer als der Open-Preis erhalten, dann ist die Reihenfolge der Trades umgekehrt. Verarbeiten nun einen 1-Minute-Balken in diesem "Cheat"-Modus, und warten auf den nächsten. Bei der Prüfung eines solchen Expert Advisors auf die Geschichte läuft alles gut, aber wenn er online läuft, sehen wir die Wahrheit - die Linie der Balance ist immer noch glatt, aber es geht nach unten. Um diesen Trick zu entlarven, müssen wir einfach den EA in der "Jeder Tick"-Modus ausführen.

**Wichtig:** Wenn die Testergebnisse von EA in den rauen Test-Modi ("1 Minute OHLC" und "Nur Eröffnungspreise") zu gut scheinen, testen Sie ihn im "Jeder Tick"-Modus.

## Nur Eröffnungspreise

In diesem Modus werden Ticks basierend auf den OHLC Preise vom Zeitraum, der für Testen ausgewählt ist, generiert. Die OnTick()-Funktion des Expert Advisors läuft nur am Anfang des Balkens auf dem Open-Preis. Aufgrund dieser Funktion Stopp-Ebenen und erledigten Aufträge können zu nicht angegebenem Preis aktiviert werden (vor allem beim Testen auf höheren Zeitrahmen). Im Gegenzug dafür sind wir in der Lage, einen Expert Advisor schnell zu testen.

Eine Ausnahme bei der Generierung von Ticks im "Nur Eröffnungspreise"-Modus ist die Perioden W1 und MN1: Zeitrahmen für diese Ticks sind für den OHLC Preise jeden Tages generiert, sondern nicht für OHLC Preise der Woche oder des Monats, jeweils.

Zum Beispiel testen wir einen Expert Advisor auf EURUSD H1 im "Nur Eröffnungspreise"-Modus. In diesem Fall ist die Gesamtzahl von Ticks (Kontrollpunkte) nicht mehr als 4\*Anzahl der 1-Stunde-Balken, die in der Testenintervall sind. Aber gleichzeitig wird OnTick()-Handler nur an der Öffnung von 1-Stunde-Balken aufgerufen. Auf der anderen ("versteckt" aus dem Expert Advisor) Ticks laufen Prüfungen, die für korrekten Testen notwendig sind:

- Berechnung der Margin-Anforderungen;
- Auslösen von Stop Loss und Take Profit Ebenen;
- Auslösen von erledigten Aufträge;
- Löschen von abgelaufenen erledigten Aufträge.

Wenn es keine offenen Positionen oder erledigten Aufträge gibt, brauchen wir nicht diese Prüfungen auf versteckten Ticks, und Performance-Gewinn kann erheblich sein. Dieser Modus "Nur Eröffnungspreise" ist gut zum Testen von Strategien, die Deals nur auf der Eröffnung von Balken machen und erledigten Aufträge nicht verwenden, und auch verwenden sie nicht StopLoss, TakeProfit. Für die Klasse solcher Strategien bleibt alle notwendigen Genauigkeit von Tests.

Verwenden wir Moving Average Expert Advisor aus dem Standard-Paket als ein Beispiel für einen EA, die in jedem Modus getestet werden kann. Die Logik dieses EAs ist so, dass alle Entscheidungen bei der Eröffnung des Balkens und Deals werden sofort ausgeführt, ohne die Verwendung von erledigten Aufträge gemacht. Starten wir Testen auf EURUSD H1 im Intervall von 2010.01.09 bis 2010.31.12 und vergleichen wir die Charts. Die Abbildung zeigt Bilanz-Charts aus dem Tester-Bericht für alle drei Modi.



Wie Sie sehen können Charts auf den verschiedenen Modi des Tests sind die gleichen für den EA Moving Average.

Es gibt mehrere Einschränkungen des Modes "Nur Eröffnungspreise":

- Sie können nicht Handelsregime "zufällige Verzögerung" verwenden;
- Im EA können Sie die Daten am Zeitraumen, der niedriger als der für die Tests/Optimierung eingesetzte Zeitraumen ist, nicht verwenden. Zum Beispiel, wenn der Test / Optimierung auf Zeitraum H1 durchgeführt wird, können Sie Zugriff auf die Daten von H2, H3, H4, etc. haben, nicht aber auf die Daten M30, M20, M10, etc. Darüber hinaus älteren Zeiträumen, die aufgerufen sind, müssen ein Vielfaches des Test-Zeitraumens sein. Zum Beispiel, wenn beim Testen auf M20 können Sie nicht M30 zugreifen, aber die Daten aus H1 sind verfügbar. Diese Einschränkungen sind aufgrund der Unfähigkeit, Daten von unteren und nicht multiplen Zeiträumen von die Balken, die während des Test/der Optimierung generiert werden, zu erhalten.
- Einschränkungen beim Zugriff auf Daten anderer Zeiträume auch auf andere Symbole, deren Daten



im Expert Advisor verwendet werden, gelten. In diesem Fall ist die Begrenzung für jedes Symbol von dem ersten Zeitrahmen abhängig, der während des Tests/der Optimierung aufgetreten ist. Z. B. die wir testen auf das Symbol und die Periode EURUSD H1, der EA greift Daten von GBPUSD M20 für das erste Mal. In diesem Fall ist der EA in der Lage, weitere Daten EURUSD H1, H2, usw. zu verwenden, sowie GBPUSD M20, H1, H2 usw.

**Wichtig:** "Nur Eröffnungspreise"-Modus ist der schnellste, aber er kann nicht für alle Trading-Strategien verwendet sein. Wählen Sie den gewünschten Test-Modus basierend auf den Eigenschaften des Handelssystems.

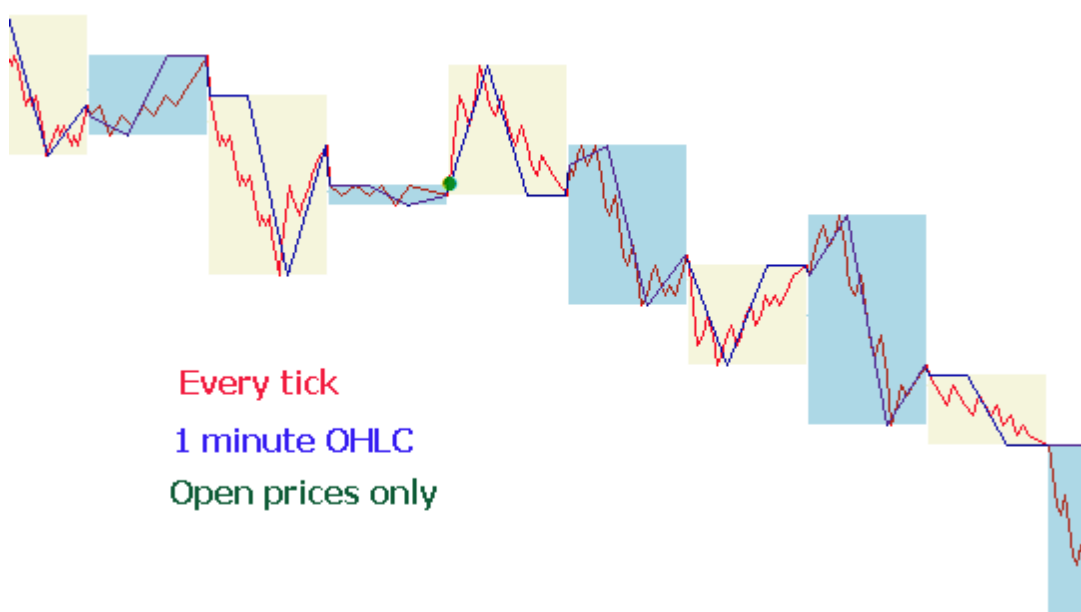
Am Ende des Abschnitts über die Modellierung Modi geben wir einen visuellen Vergleich der verschiedenen Arten der Generierung von Ticks für EURUSD für zwei Balken M15 auf dem Intervall 2011.01.11 21:00:00 - 2011.01.11 21:30:00. Die Ticks wurden in verschiedene Dateien mit Hilfe vom Expert Advisor WriteTicksFromTester.mq5 gespeichert, und das Ende der Namen dieser Dateien sind in [input-Parametern](#) filenameEveryTick, filenameOHLC und filenameOpenPrice angegeben.

| Variable                                              | Value               |
|-------------------------------------------------------|---------------------|
| <input type="checkbox"/> start                        | 2011.01.11 21:00:00 |
| <input type="checkbox"/> end                          | 2011.01.11 21:30:00 |
| <input checked="" type="checkbox"/> filenameEveryTick | everytick.csv       |
| <input checked="" type="checkbox"/> filenameOHLC      | ohlc.csv            |
| <input checked="" type="checkbox"/> filenameOpenPrice | openprice.csv       |

Strategy Tester

Settings Inputs Agents Journal

Um drei Dateien mit drei Tick-Sequenzen (für jeden der Modi "Jeder Tick", "1 Minute OHLC" und "Nur Eröffnungspreise") zu erhalten, wurde der Expert Advisor dreimal in den entsprechenden Modi mit der einzelnen Läufe gestartet. Dann wurden die Daten aus diesen drei Dateien mit Indikator TicksFromTester.mq5 auf dem Chart gezeigt. Indikator-Code wird an diesem Artikel angebracht.



Standardmäßig werden alle [Dateioperationen](#) in MQL5 Sprache innerhalb von "Datei-Sandbox" gemacht, und beim Testen kann der Expert Advisor nur eigene "Dateisystem-Sandbox" zugreifen. Um der Indikator und der EA beim Test mit Dateien von einem Ordner arbeiten könnten, verwendeten wir die Flagge [FILE\\_COMMON](#). Ein Beispiel-Code aus dem EA:

```
//--- Datei öffnen
file=FileOpen(filename,FILE_WRITE|FILE_CSV|FILE_COMMON,";");
//--- Den Erfolg der Operation überprüfen
if(file==INVALID_HANDLE)
{
    PrintFormat("Datei %s konnte nicht für Schreiben geöffnet werden. Fehlercode=%d",filename,GetLastError());
    return;
}
else
{
    //--- Informieren wir über das Schreiben in den Ordner von allen Kundenterminals
    PrintFormat("Die Datei wird im Ordner %s geschrieben",TerminalInfoString(TERMINAL_PATH));
}
```

Im Indikator beim Datenlesen haben wir auch [Flagge FILE\\_COMMON](#) verwendet. Dies erlaubt uns, manuelle Übertragung der notwendigen Dateien von einem Ordner zum anderen zu vermeiden.

```
//--- Datei öffnen
int file=FileOpen(fname,FILE_READ|FILE_CSV|FILE_COMMON,";");
//--- Den Erfolg der Operation überprüfen
if(file==INVALID_HANDLE)
{
    PrintFormat("Datei %s konnte nicht für Lesen geöffnet werden. Fehlercode=%d",fname,GetLastError());
    return;
}
else
{
    //--- Informieren über den Speicherort des freigegebenen Ordners von allen Kunden
    PrintFormat("Die Datei wird aus dem Ordner %s gelesen werden",TerminalInfoString(TERMINAL_PATH));
}
```

## Simulation von Spread

Die Differenz zwischen den Preisen Bid und Ask wird Spread genannt. Beim Testen wird der Spread nicht modelliert, sondern wird er aus historischen Daten genommen. If the spread is less than or equal to zero in the historical data, then the last known (at the moment of generation) spread of is used by testing agent.

Im Tester ist Spread immer schwimmend. D.h. [SymbolInfoInteger\(symbol, SYMBOL\\_SPREAD\\_FLOAT\)](#) gibt immer true zurück.

Darüber hinaus werden in historische Daten Werte von Tick- und Handelsvolumen gespeichert. Für das Speichern und Abrufen von Daten verwenden wir eine spezielle [MqlRates](#):

```

struct MqlRates
{
    datetime time;           // Balken Öffnungszeit
    double   open;           // Open-Preis
    double   high;           // High-Preis
    double   low;            // Low-Preis
    double   close;          // Close-Preis
    long     tick_volume;    // Tick-Volumen
    int      spread;         // Spread
    long     real_volume;    // Markt-Volumen
};

```

## Globale Variablen des Kundenterminals

Beim Testen werden [globale Variablen, des Kundenterminals](#) auch emuliert, aber sie werden nicht mit echten [globalen Variablen des Terminals](#), die können im Terminal auf der Taste F3 gesehen werden, verbunden. Dies bedeutet, dass alle Operationen mit globalen Variablen beim Testen ausserhalb des Terminals hergestellt werden (in der Test-Agent).

## Berechnung von Indikatoren beim Testen

Im Echtzeit-Modus werden die Indikator-Werte bei jedem Tick berechnet. Der Strategie-Tester hat eine kostengünstige Modell für die Berechnung der Indikatoren - [Indikatoren werden neu berechnet](#) erst unmittelbar vor dem Start von EA. Es bedeutet, dass die Neuberechnung der Indikatoren vor dem Aufruf von `OnTick()`, `OnTrade()` und `OnTimer()` durchgeführt wird.

Es spielt keine Rolle, ob es einen Aufruf für den Indikator in einem bestimmten Event-Handler gibt, werden alle Indikatoren, deren Handles durch [iCustom\(\)](#) oder [IndicatorCreate\(\)](#) erstellt wurden, vor dem Aufruf von Event-Handler neu berechnet.

Deshalb, beim Testen in "jedem Tick" werden Indikatoren vor jedem Aufruf von [OnTick \(\)](#) neu berechnet. Wenn ein Timer im EA ist durch [EventSetTimer\(\)](#)-Funktion aktiviert, werden Indikatoren vor jedem Aufruf von [OnTimer\(\)](#)-Handler neu berechnet. Daher kann die Testzeit stark mit der Verwendung eines Indikatoren, der in einer nicht optimalen Weise geschrieben ist, erhöht werden.

## Herunterladen von Geschichte beim Testen

Die Geschichte eines getesteten Symbols wird vor dem Testprozess synchronisiert und durch das Terminal aus dem Handel-Server geladen. In der ersten Zeit, lädt das Terminal alle verfügbaren Geschichte eines Symbols. Weiterhin werden nur die neuen Daten geladen.

Test-Agent erhält Geschichte des getesteten Symbols vom Terminal unmittelbar nach dem Start von Testen. Wenn Daten von anderen Instrumenten in den Testprozess verwendet werden (zum Beispiel, es ist ein Multi-Currency-Expert Advisor), dann erfordert der Test-Agent Geschichte aus dem Terminal beim ersten Anruf zu solchen Daten. Wenn historische Daten zur Verfügung am Terminal sind, wurden sie sofort an die Test-Agenten übertragen. Wenn Daten nicht verfügbar sind, lädt das Terminal sie vom Server und dann übergibt sie an Agenten.

Daten von zusätzlichen Instrumenten werden auch zur Berechnung von Cross-Rates für den Handelsoperationen erforderlich. Zum Beispiel, wenn beim Testen einer Strategie auf EURCHF mit Depot-Währung in USD, vor der Verarbeitung des ersten Handeloperation fordert der Agent historische Daten von EURUSD und USDCHF aus dem Terminal, obwohl die Strategie enthält keine direkten Aufruf

von diesen Symbolen.

Vor dem Testen einer Multi-Currency-Strategie, empfiehlt es sich, alle notwendigen historischen Daten auf dem Terminal herunter zu laden. Dies wird dazu beitragen, Verzögerungen von Tests/Optimierung auf Grund des Download der erforderlichen Daten zu vermeiden. Zum Beispiel können Sie die Geschichte durch das Öffnen des entsprechenden Charts und Scrollen ihn zu Beginn der Geschichte herunterladen. Ein Beispiel für gezwungenen Download der Geschichte in Terminal ist in der MQL5 Dokumentationabschnitt [Datenzugriffsanordnung](#) gegeben.

Test-Agenten empfangen Geschichte vom Terminal in der verpackten Form. Während des nächsten Testen, lädt Tester nicht Geschichte aus dem Terminal, weil die erforderlichen Daten seit dem letzten Lauf des Testers verfügbar sind.

- Das Terminal lädt die Geschichte aus dem Handelsserver nur einmal, wenn der Agent die Geschichte für das testierte Symbol aus dem Terminal zum ersten Mal anfordert. Die Geschichte wird in gepackter Form geladen, um den Verkehr zu reduzieren.
- Ticks werden nicht über das Netzwerk gesendet, werden sie auf Tester-Agenten generiert.

## Multi-Currency-Testen

Mit dem Tester können Sie Strategien, die auf mehreren Instrumenten handeln, auf der Geschichte überprüfen. Diese Experten werden herkömmlicherweise als Multi-Währungs-EA bezeichnet, da in den vorherigen Plattformen Test für nur ein Instrument durchgeführt wurde. MetaTrader 5 Tester kann auch Handel über alle verfügbaren Instrumente simulieren.

Geschichte der verwendeten Instrumente wird vom Tester aus dem **Kundenterminal** (nicht aus dem Handelsserver!) automatisch beim ersten Aufruf an das Instrument heruntergeladen.

Der Agent lädt nur fehlende Geschichte mit einer geringen Marge, um die notwendigen Daten über die Geschichte für die Berechnung von Indikatoren zu Beginn des Tests zu haben. Die minimale Geschichten beim Download aus dem Handelsserver für Zeiträumen D1 und weniger ist ein Jahr. Wenn daher das Testen auf dem Intervall 2010.11.01-2010.12.01 (mit jeweils einem Monat als Intervall) mit der Periode von M15 (jede Bar ist gleich 15 Minuten) gestartet wird, wird die Geschichte des Instrumentes von dem Terminal für das gesamte Jahr 2010 angefordert. Für den Zeiträumen Weekly wird die Geschichte von 100 Bars angefordert, die ungefähr zwei Jahre ist (52 Wochen pro Jahr). Für das Testen auf einer monatlichen Zeiträumen Monthly wird der Agent die Geschichte für 8 Jahre (12 Monate \* 8 Jahre = 96 Jahre) anfordern.

Wenn aus irgendwelchen Gründen die für das Testen notwendige Reserve der Bars vor dem Anfang des Testens nicht versorgt werden kann, wird das **Anfangsdatum des Testens** von der Vergangenheit bis zur Gegenwart automatisch geschoben, um solche Reserve zu gewährleisten.

Beim Testen wird auch "[Markt Durchsicht](#)" emuliert, in der man die [Informationen über Instrumente erhalten kann](#). Standardmäßig gibt es am Anfang des Testens nur ein Symbol im "Markt Durchsicht" Tester - das Symbol, auf dem das Testen gestartet wurde. Alle notwendigen Symbole werden zum "Markt Durchsicht" Tester (nicht Terminal!) automatisch beim Zugriff verbunden.

Vor dem Anfang des Testens eines Multi-Currency-Expert Advisors ist es notwendig die für das Testen erforderlichen Instrumente im " Markt Durchsicht" des Terminals auszuwählen und [die erforderlichen Daten](#) zu laden. Wenn Sie zum ersten Mal auf das fremde Symbol zugreifen, wird Daten dieses Symbols zwischen dem Test-Agenten und dem Kundenterminal automatisch synchronisiert. Das "fremde" Symbol - ist ein Symbol, das anders ist als dasjenige, auf dem das Testen gestartet wurde.

Der Zugriff auf das Daten des fremden Symbols erfolgt in den folgenden Fällen:

- Verwendung [der Funktionen der technischen Indikatoren](#) und [IndicatorCreate\(\)](#) auf Symbol/Zeitraumen;

- Anforderung an "Markt Durchsicht" (Market Watch) nach fremdem Symbol:

1. [SeriesInfoInteger](#)
2. [Bars](#)
3. [SymbolSelect](#)
4. [SymbolsSynchronized](#)
5. [SymbolInfoDouble](#)
6. [SymbolInfoInteger](#)
7. [SymbolInfoString](#)
8. [SymbolInfoTick](#)
9. [SymbolInfoSessionQuote](#)
10. [SymbolInfoSessionTrade](#)
11. [MarketBookAdd](#)
12. [MarketBookGet](#)

- Anforderung an Zeitreihen nach Symbol/Zeitraumen durch die Funktionen:

1. [CopyBuffer](#)
2. [CopyRates](#)
3. [CopyTime](#)
4. [CopyOpen](#)
5. [CopyHigh](#)
6. [CopyLow](#)
7. [CopyClose](#)
8. [CopyTickVolume](#)
9. [CopyRealVolume](#)
10. [CopySpread](#)

Wenn Sie zum ersten Mal auf das fremde Symbol zugreifen, wird der Prozess des Testens gestoppt und die Geschichte von Symbol/Zeitraumen vom Terminal auf dem Test-Agenten heruntergeladen. Gleichzeitig wird die Generierung einer Folge von Ticks für dieses Symbol aktiviert.

Für jedes Instrument wird seine eigene Folge von Ticks in Übereinstimmung mit dem gewählten Modus der Generierung von Ticks generiert. Ausserdem können Sie auch die Geschichte für die notwendigen Symbolen durch den Aufruf von der Funktion [SymbolSelect \(\)](#) im Handler OnInit () anfordern - Die

Geschichte wird unmittelbar vor dem Testen des Expert Advisors geladen.

Um das Multi-Currency-Testen im Kundenterminal MetaTrader 5 zu durchführen, ist es nicht erforderlich zusätzliche Anstrengungen zu unternehmen. Wir müssen nur die Charts der entsprechenden Instrumente im Kundenterminal öffnen. Die Geschichte der notwendigen Symbole wird automatisch vom Handels-Server geladen, sofern die Daten darauf sind.

## Simulation der Zeit im Tester

Beim Testen ist die lokale Zeit [TimeLocal\(\)](#) immer gleich der Server-Zeit [TimeTradeServer\(\)](#). Im Gegenzug ist die Server-Zeit immer gleich der Zeit, die der GMT - [TimeGMT\(\)](#) entspricht. Somit zeigen alle diese Funktionen beim Testen die gleiche Zeit.

Das Fehlen der Unterschied zwischen GMT, lokale und Server Zeit wurde in dem Tester bewusst gemacht, weil die Verbindung zum Server nicht immer hergestellt werden kann. Und die Ergebnisse des Testens sollen identisch sein, unabhängig davon, ob es die Verbindung gibt oder nicht. Information über die Zeit der Server wird nicht lokal aufbewahren, sondern wird vom Server geholt.

## Grafische Objekte beim Testen

Während des Testens / Optimierung wird die Konstruktion von grafischen Objekten nicht ausgeführt. Wenn man auf die Eigenschaften des geschaffenen Objektes während des Testens / Optimierung zugreift, erhält der Expert Advisor einen Nullwert.

Diese Einschränkung gilt nicht für das Testen im visuellen Modus.

## OnTimer()-Funktion im Tester

MQL5 bietet die Möglichkeit die Ereignisse des Timers zu bearbeiten. Der Handler [OnTimer\(\)](#) wird unabhängig von Modus des Testens aufgerufen. Dies bedeutet, dass, wenn das Testen im Modus "Nur Eröffnungspreise" auf den Zeitraum H4 gestartet wird und innerhalb des Expert Advisors wird einen Timer gestellt, der jede Sekunde aufruft, dann wird der Handler onTick () auf der Eröffnung jedes H4 Balkens einmal aufgerufen und der Handler OnTimer () wird während des Balkens 14.400-mal (3600 Sekunden \* 4.00 Stunden ) aufgerufen. Soviel die Zeit des Testens des Expert Advisors dabei zunehmen wird, hängt von der Logik des Expert Advisors ab.

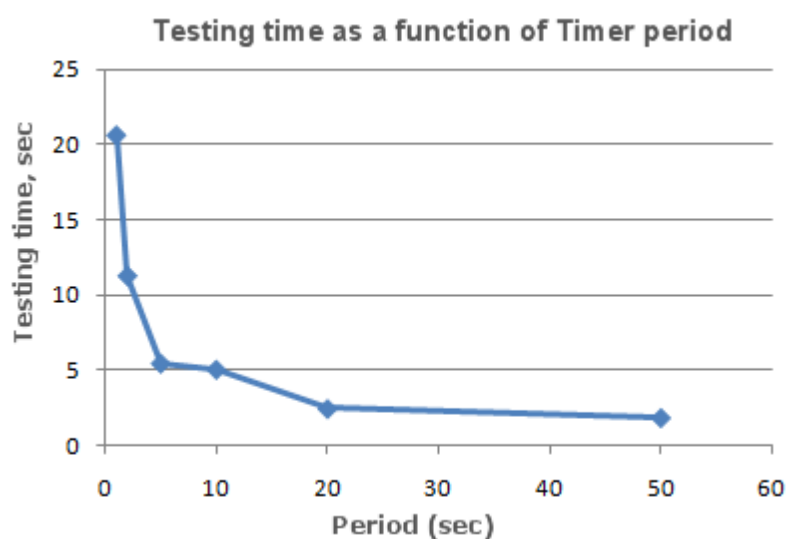
Um die Abhängigkeit der Zeit des Testens von der vorgegebenen Frequenz des Timers zu überprüfen, wurde ein einfacher Expert Advisor ohne Handelsoperationen geschrieben.

```

//--- input parameters
input int      timer=1;           // Timerwert, Sek
input bool     timer_switch_on=true; // Timer ist aktiviert
//+-----+
//| Expert initialization function |
//+-----+
int OnInit()
{
    //--- Timer läuft wenn timer_switch_on==true
    if(timer_switch_on)
    {
        EventSetTimer(timer);
    }
    //---
    return(INIT_SUCCEEDED);
}
//+-----+
//| Expert deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
    //--- stoppen den Timer
    EventKillTimer();
}
//+-----+
//| Timer function |
//+-----+
void OnTimer()
{
    //---
    // unternehmen wir nichts, der Körper des Handlers ist leer
}
//+-----+

```

Die Zeit des Testens wurde bei verschiedenen Werten des Parameter von Timer (Periodizität des Ereignisses Timer) gemessen. Auf den erhaltenen Daten ist der Chart der Abhängigkeit der Zeit des Testens T von der Wert der Periodizität Period gebaut.



Es ist deutlich zu erkennen, je geringer der Parameter Timer bei der Initialisierung durch Funktion [EventSetTimer](#)(Timer) ist, desto weniger ist der Zeitraum (Period) zwischen den Aufrufen der Handler

OnTimer (), und desto mehr ist die Zeit des Testens T unter den gleichen anderen Bedingungen.

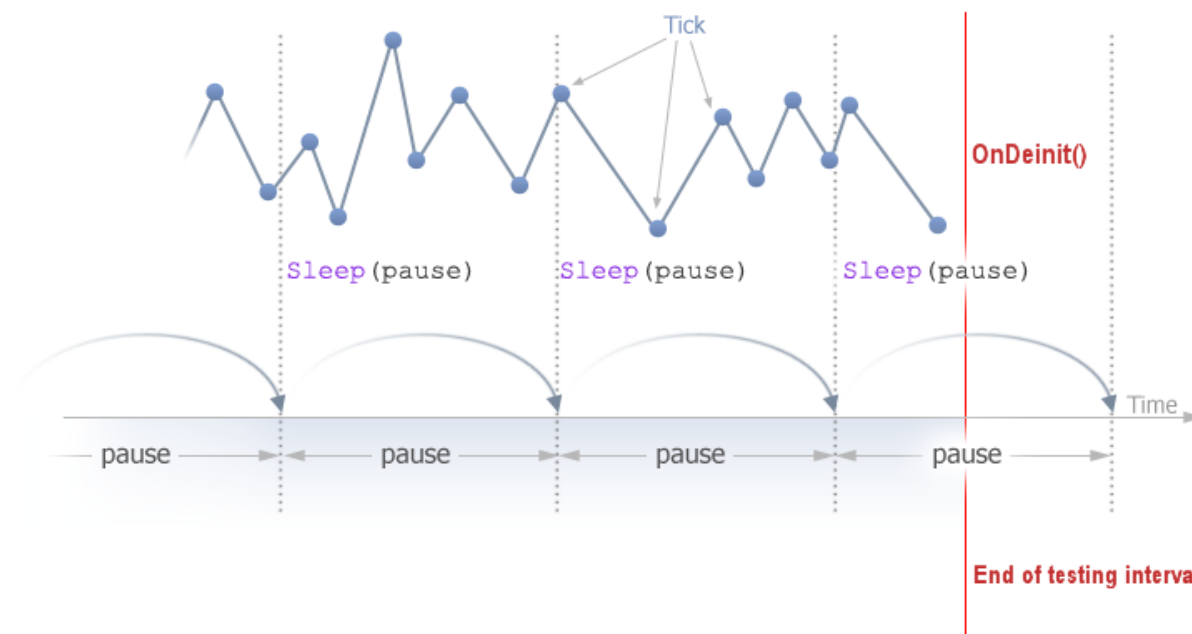
## Sleep()-Funktion im Tester

Die Funktion [Sleep\(\)](#) ermöglicht es Ihnen, im Expert Advisor oder Script die Ausführung des mql5-Programms für eine Zeitlang bei der Arbeit auf dem Chart auszusetzen. Dies kann nützlich sein wenn Sie irgendwelche Daten, die zum Zeitpunkt der Anfrage noch nicht verfügbar sind, anfordern und Sie müssen warten, bis sie bereit sind. Das ausführliche Beispiel der Nutzung der Funktion Sleep () können Sie im Abschnitt [Datenzugriff organisieren](#) finden.

Im Tester halten die Aufrufe Sleep () den Prozess des Testens nicht auf. Beim Aufruf von Sleep() werden die generierten Ticks innerhalb der angegebenen Verzögerung "gespielt", infolgedessen können offene Aufträge, Stops usw. auslösen. Nach dem Aufruf Sleep () verlängert sich die im Tester simulierte Zeit in einem Intervall, das im Parameter der Funktion Sleep angegeben ist.

Wenn als Ergebnis der Ausführung der Funktion Sleep () die aktuelle Zeit im Tester über den Zeitraum des Testens hinausgeht, dann erhalten Sie eine Fehlermeldung ' Endlosschleife in Sleep '. Wenn Sie diese Fehlermeldung erhalten, die Ergebnisse des Testens nicht abgelehnt werden. Alle Berechnungen werden in ihrem vollen Umfang (die Anzahl der Deals, Drawdown, etc.) durchgeführt und die Ergebnisse dieses Tests werden auf dem Terminal weitergegeben.

Die Funktion Sleep () wird in OnDeinit () nicht funktionieren, da nach ihrem Aufruf die Zeit des Testens das Intervall des Testens garantiert übertreffen wird .



## Verwenden des Testers für Optimierungsprobleme in mathematischen Berechnungen

Der Tester im MetaTrader 5-Terminal kann nicht nur für das Testen der Handelsstrategien, sondern auch für mathematische Berechnungen verwendet werden. Dafür muss man in den Einstellungen den entsprechenden Modus wählen:



Wenn Sie den Modus "Mathematische Berechnungen" wählen, wird der Test-Agent "leer" laufen. Der leerer Lauf bedeutet, dass Generierung der Ticks und Herunterladen der Geschichte nicht durchgeführt werden. Bei solchem Lauf werden nur drei Funktionen aufgerufen: OnInit(), OnTester(), OnDeinit().

Wenn das Enddatum des Testens kleiner oder gleich dem Anfangsdatum des Testens ist, so bedeutet es, dass die Testens im Modus "Mathematische Berechnungen" durchgeführt werden.

Wenn der Tester für Lösung der mathematischen Aufgaben verwendet wird, wird die Geschichte nicht heruntergeladen und die Ticks werden nicht generiert.

Eine typische mathematische Aufgabe für die Lösung im Tester MetaTrader 5 - die Suche nach einem Extremum einer Funktion mit vielen Variablen. Um sie zu lösen, müssen Sie folgendes tun:

- Legen Sie einen Berechnungsblock der Funktionswerten von mehreren Variablen in [OnTester\(\)](#) und geben Sie den berechneten Wert durch `return(Wert_der Funktion)` zurück;
- Übertragen Sie die Parameter der Funktion auf die globale Reichweite des Programms als [input-Variablen](#);

Kompilieren wir den Expert Advisor, öffnen wir das Fenster "Tester". Auf der Registerkarte "Eingabeparameter" wählen wir die erforderlichen Input Variablen und definieren wir für sie die Reihe von Parameterwerten und den Schritt für das Durchprobieren.

Wählen wir den Typ der Optimierung - "Langsam( vollständige Suche der Parameter)" oder - "Schnell (genetischer Algorithmus)". Für eine einfache Suche des Extremums der Funktion ist es besser die schnelle Optimierung zu wählen, aber wenn man die Werte für die ganze Reihe von Variablen ausrechnen muss, dann ist es am besten die langsame Optimierung zu verwenden.

Wählen wir den Modus "Mathematische Berechnungen" und klicken wir auf "Start" um den Optimierungsprozess zu starten. Beachten Sie, dass bei der Optimierung immer das lokale Maximum des Wertes von der Funktion OnTester gesucht wird. Für die Suche lokalen Minimums kann man das Inverse des ausgerechneten Wertes der Funktion aus der Funktion OnTester zurückgeben:

```
return(1/Wert_der Funktion);
```

Dabei muss man selbständig überprüfen, dass der Wert\_der Funktion null nicht gleich ist, sonst man [kritischen Fehler](#) der Division durch Null erhalten kann. Es gibt noch eine andere Option, die bequemer ist und verzerrt die Ergebnisse der Optimierung nicht, sie wurde von den Lesern des Artikels vorgeschlagen:

```
return(-Wert_der Funktion);
```

In dieser Option muss man der Wert\_der Funktion auf Gleichheit mit Null nicht überprüfen und die Oberfläche der Optimierungsergebnisse in 3D-Darstellung selbst hat die gleiche Form, die nur von der ersten Option gespiegelt ist.

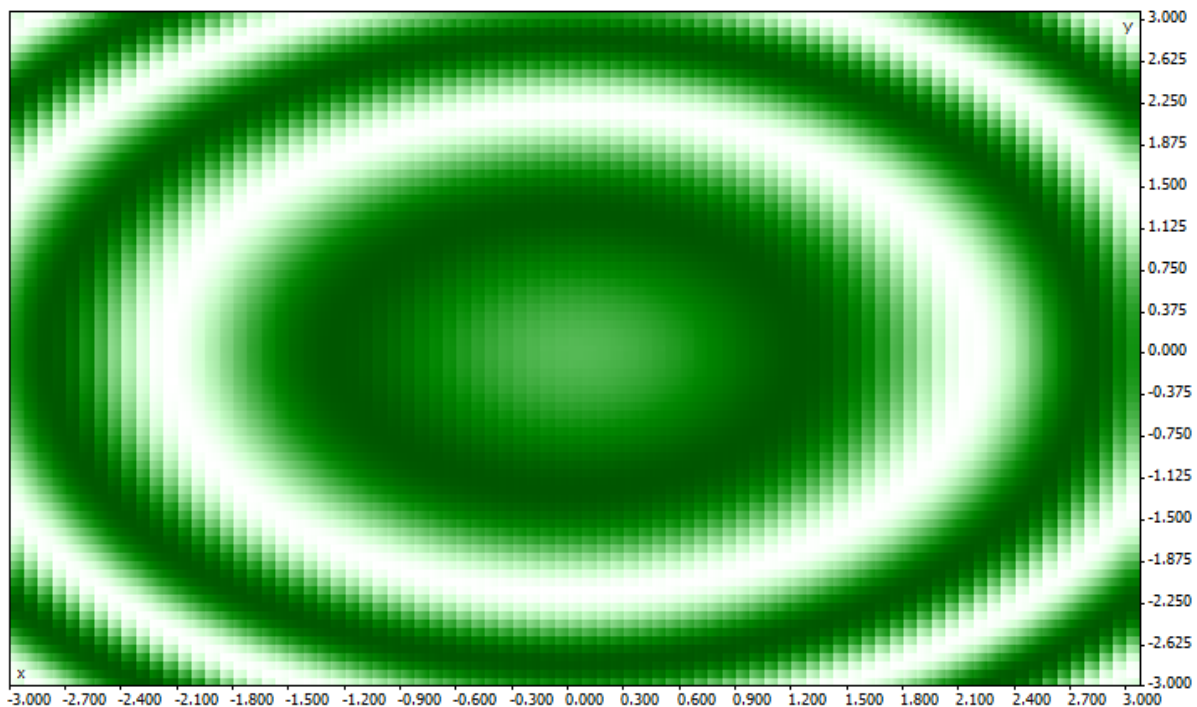
Verwenden wir die Funktion `sink()` als Beispiel:

$$\text{sink}(x,y) = \sin(x^2 + y^2)$$

Stellen wir den Code des Expert Advisors für die Suche des Extremums dieser Funktion in `OnTester()`:

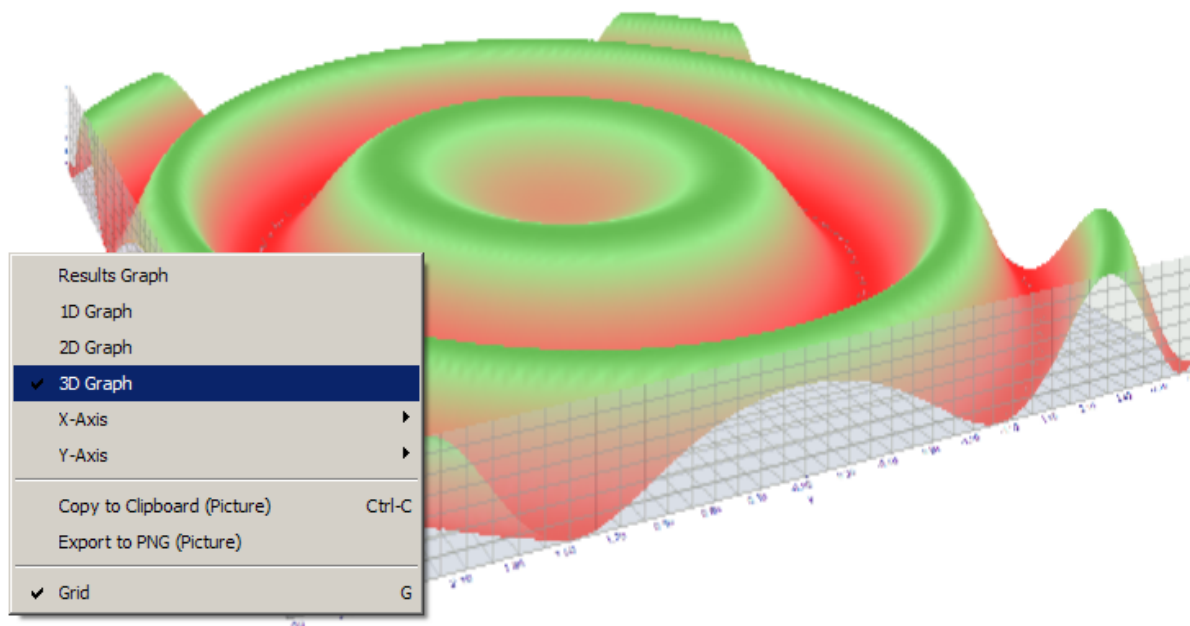
```
//+-----+
//|                                     Sink.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- input parameters
input double   x=-3.0; // start=-3, step=0.05, stop=3
input double   y=-3.0; // start=-3, step=0.05, stop=3
//+-----+
//| Tester function |
//+-----+
double OnTester()
{
//---
    double sink=MathSin(x*x+y*y);
//---
    return(sink);
}
//+-----+
```

Durchführen wir eine Optimierung und stellen wir die [Ergebnisse der Optimierung](#) in Form von 2D Chart vor.



Je besser der Wert für ein angegebenes Paar von Parametern (x, y) ist, desto gesättigter ist die Farbe. Wie es aus der Sicht der Formel der Funktion `sink()` erwartet wurde, deren Wert konzentrische

Kreise mit Mittelpunkt bei (0,0) bildet. Für die Funktion  $\sin(x)$  existiert kein absolutes Extremum. Es ist bei der Anzeige der Ergebnisse der Optimierung im 3D-Modus gut sichtbar:



## Synchronization von Balken beim Testen in "Open Prices Only"-Modus

Der Tester im MetaTrader 5 Kundenterminal ermöglicht es Ihnen so genannte "Multi-Currency" Expert Advisors zu überprüfen. Ein Multi-Currency-Expert Advisor ist ein Expert Advisor, der auf zwei oder mehr Symbole handelt.

Das Testen der Strategien, die auf einigen Instrumenten handeln, legt einige zusätzliche technische Anforderungen an den Tester:

- Die Generierung der Ticks für diese Instrumenten;
- Die Berechnung der Indikatorwerte für diese Instrumenten;
- Berechnung der Margin-Anforderungen für diese Instrumenten;
- Die Synchronization der generierten Ticksfolgen für alle Handelsinstrumenten.

Der Tester generiert und spielt eine Tick-Sequenz für jedes Instrument in Übereinstimmung mit dem ausgewählten Modus des Handels. Dabei wird eine neue Bar für jedes Instrument geöffnet, unabhängig davon, wie die Bar auf anderem Instrument geöffnet wurde. Dies bedeutet, dass beim Testen des Multi-Currency-Expert Advisors die Situation (meistens ist es der Fall) auftreten kann, wenn eine neue Bar für ein Instrument bereits geöffnet wurde und noch nicht für die anderen. So passiert alles beim Testen genauso wie im wirklichen Leben.

Solche authentische Simulation der Geschichte im Tester verursacht keine Probleme, solange die Modi "Jeder Tick" und "1 Minute OHLC" des Testens verwendet werden. Bei diesen Modi wird eine ausreichende Anzahl von Ticks innerhalb einer Kerze generiert, um den Moment der Synchronisation der Bars von verschiedenen Symbolen erwarten zu können. Aber wie kann man die Multi-Currency-Strategien im Modus "Nur Eröffnungspreise" testen, wenn die Bars auf Handelsinstrumente zwingend synchronisiert werden muss? In diesem Modus wird der Expert Advisor nur auf einen Tick aufgerufen, der der Öffnungszeit der Bar entspricht.

Wir erklären nun am Beispiel: Wenn wir den Expert Advisor auf das Symbol EURUSD testen, und auf EURUSD wurde eine neue stündliche Kerze geöffnet, dann erfahren wir leicht diese Tatsache - beim Testen im Modus "Nur Eröffnungspreise" entspricht das Ereignis [NewTick](#) dem Moment der Eröffnung der Bar auf den Zeitraum des Testens. Aber dabei gibt es keine Garantie, dass die neue Kerze nach dem Symbol GBPUSD geöffnet wurde, das im Expert Advisor verwendet wird.

Unter gewöhnlichen Umständen ist es ausreichend genug die Arbeit der Funktion [OnTick\(\)](#) zu beenden und das Erscheinen einer neuen Bar auf USDJPY auf dem nächsten Tick zu überprüfen. Aber beim Testen im Modus "Nur Eröffnungspreise" wird es kein anderer Tick sein, und so mag es scheinen, dass dieses Modus für das Testen des Currency-Expert Advisors nicht passt. Aber das ist nicht so - vergessen Sie nicht, dass sich der Tester in MetaTrader 5 genauso wie im wirklichen Leben verhält. Sie können warten, bis eine neue Bar auf einem anderen Symbole mittels Funktion [Sleep\(\)](#) geöffnet wird!

Der Code des Expert Advisors [Synchronize\\_Bars\\_Use\\_Sleep.mq5](#), der ein Beispiel der Synchronisation der Bars beim Testen im Modus "Nur Eröffnungspreise" zeigt.

```

//+-----+
//|                                     Synchronize_Bars_Use_Sleep.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- input parameters
input string  other_symbol="USDJPY";
//+-----+
//| Expert initialization function |
//+-----+
int OnInit()
{
//--- nachprüfen wir das aktuelle Symbol
if(_Symbol==other_symbol)
{
PrintFormat("Man muss ein anderes Symbol angeben oder starten Sie das Testen auf %s", other_symbol);
//--- stoppen wir das Testen des Expert Advisors zwangsmäßig
return(INIT_PARAMETERS_INCORRECT);
}
//---
return(INIT_SUCCEEDED);
}
//+-----+
//| Expert tick function |
//+-----+
void OnTick()
{
//--- statistische Variable für die Speicherung der Zeit der Öffnung der letzten Bar
static datetime last_bar_time=0;
//--- das Merkmal der Synchronisation der Öffnungszeit der letzten Bar auf verschiedene Symbole
static bool synchronized=false;
//--- wenn statische Variable ist nicht initialisiert
if(last_bar_time==0)
{
//--- es ist ein erster Aufruf, speichern wir die Öffnungszeit und verlassen ihn
last_bar_time=(datetime)SeriesInfoInteger(_Symbol,Period(),SERIES_LASTBAR_DATE);
PrintFormat("Variable wird mit dem Wert %s initialisiert",TimeToString(last_bar_time));
}
//--- erhalten wir die Öffnungszeit der letzten Bar nach unserem Symbol
datetime curr_time=(datetime)SeriesInfoInteger(Symbol(),Period(),SERIES_LASTBAR_DATE);
//--- wenn die Zeit der Öffnung der aktuellen Bar ist nicht gleich der Zeit, die gespeichert ist
if(curr_time!=last_bar_time)
{
//--- speichern wir die Öffnungszeit der neuen Bar in statistischer Variable
last_bar_time=curr_time;
//--- nicht synchronisiert
synchronized=false;
//--- geben wir die Meldung über dieses Ereignis aus
PrintFormat("Auf dem Symbol %s wurde eine neue Bar in %s geöffnet",_Symbol,TimeToString(last_bar_time));
}
//--- hier werden wir die Öffnungszeit der Bar auf dem fremden Symbol speichern
datetime other_time;
//--- Schleife bis die Öffnungszeit des letzten Bars nach anderem Symbol wird gleich der Zeit, die gespeichert ist
while(!(curr_time==(other_time=(datetime)SeriesInfoInteger(other_symbol,Period(),SERIES_LASTBAR_DATE))))
{
PrintFormat("Warten Sie 5 Sekunden..");
//--- warten Sie 5 Sekunden und anfordern Sie noch einmal SeriesInfoInteger(other_symbol,Period(),SERIES_LASTBAR_DATE)
Sleep(5000);
}
}

```

```

    }
    //--- die Öffnungszeit der Bar ist gleich für beide Symbole
    synchronized=true;
    PrintFormat("Die Öffnungszeit der letzten Bar auf dem Symbol %s ist: %s",_Symbol,TimeCurrent());
    PrintFormat("Die Öffnungszeit der letzten Bar auf dem Symbol %s ist: %s",other_symbol,TimeCurrent());
    //--- TimeCurrent() passt nicht, verwenden wir TimeTradeServer() für
    Print("Die Bars wurden synchronisiert um",TimeToString(TimeTradeServer()),TIME_DATE|TIME_SECONDS);
    }
    //+-----+

```

Beachten Sie die letzte Zeile im Handler, die die aktuelle Zeit anzeigt, wenn die Tatsache der Synchronisation festgestellt wurde:

```
Print("Die Bars wurden synchronisiert um ",TimeToString(TimeTradeServer()),TIME_SECONDS);
```

Um die aktuelle Uhrzeit anzuzeigen, verwendeten wir die Funktion [TimeTradeServer\(\)](#), und nicht [TimeCurrent\(\)](#). Es geht darum, dass die Funktion TimeCurrent () die Zeit des letzten Ticks zurückgibt, die sich nach der Verwendung von Sleep() auf keine Weise geändert hat. Starten Sie den Handler im Modus "Nur Eröffnungspreise" und Sie erhalten die Meldungen über Synchronisation der Bars.

|        |                     |                                                            |
|--------|---------------------|------------------------------------------------------------|
| Core 1 | 2010.12.01 20:00:05 | The bars are synchronized at 2010.12.01 20:00:05           |
| Core 1 | 2010.12.01 20:00:05 | Open bar time of the chart symbol USDJPY: 2010.12.01 20:00 |
| Core 1 | 2010.12.01 20:00:05 | A new bar has appeared on symbol EURUSD: 2010.12.01 20:00  |
| Core 1 | 2010.12.01 20:00:00 | Waiting 5 seconds..                                        |
| Core 1 | 2010.12.01 20:00:05 | A new bar has appeared on symbol EURUSD: 2010.12.01 20:00  |
| Core 1 | 2010.12.01 16:00:05 | The bars are synchronized at 2010.12.01 16:00:05           |

Verwenden Sie die Funktion TimeTradeServer () anstelle von TimeCurrent (), wenn Sie die aktuelle Serverzeit und nicht die Eingangszeit des letzten Ticks erhalten möchten.

Es gibt einen anderen Weg die Bars zu synchronisieren - mit Hilfe eines Timers. Ein Beispiel solchen Expert Advisors ist Synchronize\_Bars\_Use\_OnTimer.mq5, der dem Artikel beigelegt ist.

## IndicatorRelease()-Funktion im Tester

Nach dem Abschluss des einzelnen Testens wird ein Chart des Instrumentes automatisch geöffnet, auf dem die abgeschlossenen Deals und die Indikatoren angezeigt werden, die in Expert Advisor verwendet wurden. Das hilft die Momente des Eingangs und des Ausgangs optisch zu überprüfen und sie mit den Werten der Indikatoren zu vergleichen.

**Hinweis:** Die Indikatoren, die auf dem automatisch geöffneten Chart nach der Durchführung des Testens angezeigt werden, werden nach Beendigung des Testens neu berechnet. Auch wenn diese Indikatoren wurden in dem getesteten Expert-Advisor verwendet.

Aber in einigen Fällen benötigt der Programmierer die Informationen über die im Handelsalgorithmus mitwirkenden Indikatoren auszublenden. Zum Beispiel, wird der Code des Experten vermietet oder wird als Ausführungsdatei ohne Quellcode verkauft. Für diese Ziele ist die Funktion IndicatorRelease() geeignet.

Wenn im Terminal die Schablone mit dem Namen tester.tpl im Verzeichnis /profiles/templates des Kundenterminals angegeben wird, dann wird sie zum geöffneten Chart verwendet werden. In ihrer Abwesenheit wird die Standard-Schablone angewendet (default.tpl).

Die Funktion [IndicatorRelease\(\)](#) wird für die Befreiung von Berechnungen des Indikators verwendet, wenn es nicht mehr nötig ist. So können Sie sowohl Speicherplatz als auch Prozessor-Ressourcen sparen, denn jeder Tick verursacht die Berechnung des Indikators. Ihr zweiter Zweck ist das Verbot der Anzeige des Indikators auf dem Chart des Testens nach dem einzelnen Lauf.

Um die Anzeige des Indikators auf dem Chart am Ende des Testens zu verbieten, rufen Sie [IndicatorRelease\(\)](#) mit dem Handle des Indikators im Handler [OnDeinit\(\)](#). Die Funktion [OnDeinit\(\)](#) wird immer nach dem Abschluss oder vor der Anzeige des Charts des Testens aufgerufen.

```
//+-----+
//| Expert deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
//---
    bool hidden=IndicatorRelease(handle_ind);
    if(hidden) Print("IndicatorRelease() erfolgreich abgeschlossen");
    else Print("IndicatorRelease() gab false zurück. Fehlercode ",GetLastError());
}
```

Um die Anzeige des Indikators auf dem Chart nach dem Abschluss des einzelnen Testens zu verbieten, verwenden Sie die Funktion [IndicatorRelease\(\)](#) im Handler [OnDeinit\(\)](#).

## Behandlung von Ereignissen im Tester

Das Vorhandensein des Handlers [OnTick\(\)](#) im Expert Advisor ist nicht obligatorisch, um ihn der Überprüfung auf die historischen Daten im Tester des MetaTrader 5 Terminals zu unterwerfen. Es ist ausreichend für den Expert Advisor wenigstens einen der folgenden Funktion-Handlers zu haben:

- [OnTick\(\)](#) - Event-Handler des Erhaltes neuen Ticks;
- [OnTrade\(\)](#) - Handler des Handelereignisses;
- [OnTimer\(\)](#) - Event-Handler der Ankunft des Signals aus dem Timer;
- [OnChartEvent\(\)](#) - Handler der Benutzerereignisse.

Beim Testen kann man die Benutzerereignisse mit Hilfe der Funktion [OnChartEvent\(\)](#) im Expert Advisor bearbeiten, aber in den Indikatoren wird diese Funktion im Tester nicht aufgerufen. Auch wenn der Indikator den Handler [OnChartEvent\(\)](#) hat und dieser Indikator wird in dem getesteten Expert Advisor verwendet, wird der Indikator keine Benutzerereignisse bekommen.

Der Indikator kann beim Testen die Benutzerereignisse mittels Funktion [EventChartCustom\(\)](#) generieren, und der Expert Advisor kann dieses Ereignis in [OnChartEvent\(\)](#) bearbeiten.

Zusätzlich zu den obengenannten Ereignissen werden im Strategie-Tester die speziellen Ereignisse generiert, die mit dem Prozess des Testens und der Optimierung verbunden sind:

- **Tester** - dieses Ereignis wird nach dem Testen des Expert Advisors auf historische Daten generiert. Das Ereignis **Tester** wird durch die Funktion [OnTester\(\)](#) bearbeitet. Diese Funktion kann nur in Experten beim Testen verwendet werden und dient Zuerst für die Berechnung eines Wertes, der als Kriterium Custom max bei genetischer Optimierung der Eingabeparameter verwendet wird.
- **TesterInit** - dieses Ereignis wird beim Start der Optimierung im Strategie-Tester vor dem ersten Durchlauf generiert. Bearbeitung des Ereignisses **TesterInit** wird durch die Funktion [OnTesterInit\(\)](#) durchgeführt. Der Expert Advisor, der diesen Handler hat, wird beim Start der Optimierung automatisch auf einem gesonderten Chart des Terminals geladen mit dem Symbol und der Periode,

die im Tester angegeben wurden, und ein Ereignis `TesterInit` empfängt. Die Funktion soll einen Expert Advisor vor Optimierung initialisieren um dann die Ergebnisse der Optimierung zu bearbeiten.

- `TesterPass` - dieses Ereignis wird beim Empfang eines neuen Datenframes generiert. Bearbeitung des Ereignisses `TesterInit` wird durch die Funktion `OnTesterPass()` durchgeführt. Der Expert Advisor mit diesen Handler wird automatisch auf einem gesonderten Chart des Terminals geladen mit dem für ein Test angegebenen Symbol und der Periode, und erhält Ereignisse `TesterPass` beim Erhalten von einem Frame bei der Optimierung. Die Funktion wird für die dynamische Verarbeitung von Optimierungsergebnisse direkt "on the fly" ausgelegt, ohne für die Vollendung der Optimierung zu warten. Frames werden durch Funktion `FrameAdd()` hinzugefügt. Diese Funktion kann nach einem Durchgang im Handler `OnTester()` aufgerufen werden.
- `TesterDeinit` - dieses Ereignis wird nach der Optimierung eines Expert Advisors generiert. Bearbeitung des Ereignisses `TesterDeinit` wird durch die Funktion `OnTesterDeinit()` durchgeführt. Der Expert Advisor mit diesem Handler wird automatisch auf dem Chart beim Start der Optimierung geladen, und erhält ein Ereignis `TesterDeinit` nach ihrer Fertigstellung. Die Funktion wird für die abschließende Bearbeitung aller Optimierungsergebnisse verwendet.

## Testagenten

Das Testen wird mit Hilfe der Testagenten im MetaTrader 5 Kundenterminal durchgeführt. Lokalen Agenten werden automatisch erstellt und aktiviert. Die Anzahl der lokalen Objekte entspricht standardmäßig der Anzahl der Kerne auf dem Computer.

Jeder Testagent hat seine eigene Kopie von globalen Variablen, die mit dem Kundenterminal nicht verbunden ist. Das Terminal ist ein Dispatcher, der die Aufgaben an die lokalen und Remote-Agenten verteilt. Wenn Sie die nächste Aufgabe des Testens des Expert Advisors mit den angegebenen Parametern ausgeführt haben, gibt der Agent dem Terminal die Ergebnisse zurück. Beim einzelnen Testen wird nur einen Agenten verwendet.

Der Agent speichert die vom Terminal erhaltenen Geschichte in den einzelnen Ordnern mit dem Namen des Instrumentes, das heisst die Geschichte für EURUSD in einem gesonderten Ordner mit dem Namen EURUSD aufbewahrt wird. Ausserdem wird die Geschichte des Instrumentes durch die Quellen geteilt. Die Struktur für die Speicherung der Geschichte ist wie folgt:

```
Verzeichnis_des Testers\Agent-IPaddress-Port\bases\Name_der Quelle\history\Name_des I
```

Zum Beispiel kann die Geschichte der EURUSD von MetaQuotes-Demo-Server in einem Ordner `Verzeichnis_des Testers \ Agent-127.0.0.1-3000 \ bases \ MetaQuotes-Demo \ EURUSD` gespeichert werden.

Der lokale Agent befindet sich nach Beendigung des Testens im Bereitschaftsmodus und wartet auf die nächste Aufgabe im Laufe von 5 Minuten, um die Zeit mit dem Start bei den nächsten Aufrufen nicht zu verschwenden. Nur stellt nach Ablauf der Wartezeit der lokale Agent seine Arbeit ein und wird aus dem Speicher des Computers entladet.

Bei der vorzeitigen Beendigung des Testens seitens des Benutzers (Schaltfläche "Abbrechen"), und auch bei der Schließung des Kundenterminals stellen alle lokalen Agenten ihre Arbeit sofort ein und werden aus dem Speicher entladet.

## Der Datenaustausch zwischen dem Terminal und dem Agenten



Beim Start des Testens bereitet das Terminal einige Parameter-Blöcke für das Senden an den Agenten vor:

- Die Eingabeparameter des Testens (Simulationsmodus, das Intervall des Testens, Instrumente, Optimierungskriterium usw)
- Liste der im "Markt Durchsicht" ausgewählten Instrumente
- Die Spezifikation des getesteten Instrumentes (die Kontraktgröße, die zulässigen Margen vom Markt für die Einstellung von StopLoss und Takeprofit, usw)
- Der getestete Expert-Advisor und die Werte seiner Eingabeparameter
- Informationen über weitere Dateien (Bibliotheken, Indikatoren, Daten-Dateien - [#\\_Property\\_tester\\_...](#))

|                  |                        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|------------------|------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tester_indicator | <a href="#">string</a> | Name des Benutzerindikators im Format "Name_des Indicators.ex5." Die für Testen erforderliche Indikatoren werden automatisch aus Aufruf der Funktionen <a href="#">iCustom()</a> bestimmt, wenn der entsprechende Parameter von der Konstantzeile vorgegeben ist. Für andere Fälle (Verwendung der Funktion <a href="#">IndicatorCreate()</a> oder Verwendung einer nicht Konstantzeile im Parameter, der den Namen des Indikator vorgibt) ist diese Eigenschaft erforderlich |
| tester_file      | <a href="#">string</a> | Dateiname für Tester mit Angabe der Erweiterung, in Doppelanführungszeichen (als Konstantzeile). Die angegebene Datei wird dem Tester für Arbeit übertragen. Eingabeparameter für Testen müssen immer angegeben werden (wenn sie erforderlich sind).                                                                                                                                                                                                                          |
| tester_library   | <a href="#">string</a> | Name der Bibliothek mit Erweiterung in Doppelanführungszeichen. Bibliothek kann sowohl mit der Erweiterung dll, als auch mit der Erweiterung ex5 sein kann. Die für Testen erforderliche Bibliotheken werden automatisch                                                                                                                                                                                                                                                      |

|  |  |                                                                                                                        |
|--|--|------------------------------------------------------------------------------------------------------------------------|
|  |  | bestimmt. Aber wenn eine Bibliothek vom <u>Benutzerindikator</u> verwendet wird, muss man diese Eigenschaft verwenden. |
|--|--|------------------------------------------------------------------------------------------------------------------------|

Für jedes Parameter-Block wird ein digitaler Fingerabdruck in Form von MD5-Hash erstellt, der an den Agenten gesendet wird. MD5-Hash ist einzigartig für jeden Satz, sein Volumen ist viel kleiner als die Informationsmenge, auf deren Grundlage er berechnet wurde.

Der Agent erhält Hashes von Blöcken und vergleicht sie mit denen, die er bereits hat. Wenn der Fingerabdruck dieses Parameter-Blocks dem Agenten fehlt, oder der empfangene Hash unterscheidet sich von der bestehenden, anfordert der Agent selbst das Parameter-Block. So wird der Verkehr zwischen dem Terminal und dem Agenten reduziert.

Nach der Durchführung des Testens gibt der Agent alle Ergebnisse des Laufs zurück. Sie werden auf der Registerkarten "Testergebnisse" und "Optimierungsergebnisse" angezeigt: Der Gewinn, die Anzahl der Deals, die Sharpe Ratio, das Ergebnis der Funktion OnTester (), etc.

Bei der die Optimierung verteilt das Terminals die Aufgaben für die Durchführung des Testens in kleine Pakete an die Agenten, jedes Paket enthält mehrere Aufgaben (jede Aufgabe bedeutet einzelnes Testen mit einem Satz von Eingabeparametern). Dies reduziert die Zeit des Austausches zwischen dem Terminal und dem Agenten.

Agenten werden die vom Terminal erhaltenen EX5-Dateien (Expert-Advisor, Indikatoren, Bibliotheken, etc.) aus Gründen der Sicherheit nie auf Ihrer Festplatte speichern, damit man die gesendeten Daten auf einem Computer mit laufendem Agenten nicht benutzen konnte. Alle übrigen Dateien, einschließlich DLL, werden in der "Sandbox" gespeichert. In der Remote-Agenten kann man die Experten unter Verwendung der DLL nicht testen.

Die Testergebnisse werden in einem speziellen Cache der Ergebnisse (der Ergebniscache) für den nachfolgenden schnellen Zugriff darauf sie, falls notwendig, aufbewahrt. Für jeden Satz von Parametern sucht das Terminals im Ergebniscache nach bereits vorliegenden Ergebnissen von den vorherigen Starts um den Neustart zu vermeiden. Wenn das Ergebnis mit einem solchen Satz von Parametern nicht gefunden wird, wird die Aufgaben für die Durchführung des Testens dem Agenten gegeben.

Der gesamte Verkehr zwischen dem Terminal und dem Agenten wird verschlüsselt.

Ticks werden nicht über das Netzwerk gesendet, werden sie auf Tester-Agenten generiert.

## Verwendung von einen freigegebenen Ordner von allen Kundenterminals

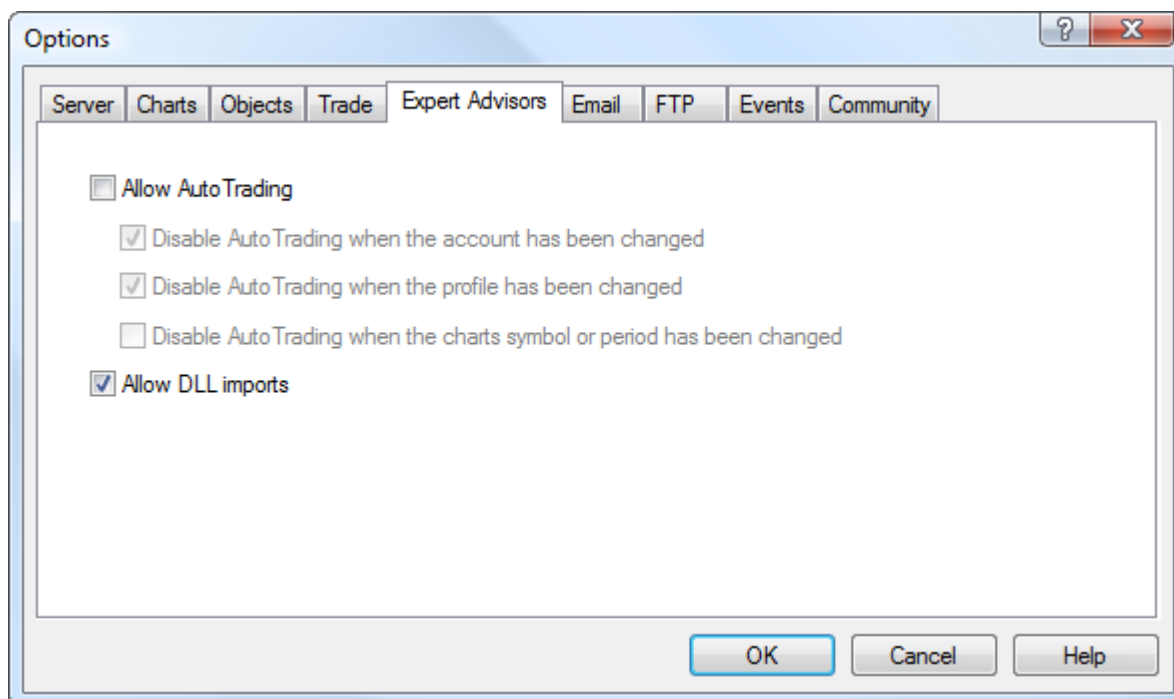
Alle Testagenten sind voneinander und von der Kundenterminal isoliert: jeder Agent hat einen eigenen Ordner, in dem seine Protokolle geschrieben werden. Ausserdem werden alle Dateioperationen beim Testen des Agenten im Ordner **Name\_des Agenten/MQL5/Files** durchgeführt. Man kann jedoch die Wechselwirkung zwischen den lokalen Agenten und dem Kundenterminal über einen freigegebenen Ordner aller Kundenterminals zu verwirklichen, wenn Sie beim Öffnen der Datei das Flag FILE\_COMMON angeben:

```
//+-----+
//| Expert initialization function |
//+-----+
int OnInit()
{
//--- der freigegebene Ordner von allen Kundenterminals
    common_folder=TerminalInfoString(TERMINAL_COMMONDATA_PATH);
//--- geben wir den Namen dieses Ordners aus
    PrintFormat("Öffnen wir die Datei in dem freigegebenen Ordner der Kundenterminals% s",
//--- öffnen wir eine Datei in dem freigegebenen Ordner (das Flag FILE_COMMON wird an
    handle=FileOpen(filename,FILE_WRITE|FILE_READ|FILE_COMMON);
    ... weitere Aktionen
//---
    return(INIT_SUCCEEDED);
};
```

## Verwendung von DLL

Zur Beschleunigung der Optimierung kann man nicht nur lokalen sondern auch [Remote-Agenten](#) verwenden. Dabei gibt es einige Beschränkungen für Remote-Agenten . Erstens können die Remote-Agenten in ihren Protokollen die Ergebnisse der Ausführung der Funktion [Print\(\)](#), Nachrichten über das Öffnen und Schließen von Positionen nicht ausgeben. Im Protokoll werden Mindestinformationen ausgegeben, so dass die falsch geschriebenen Experten mit Nachrichten die Festplatte Ihres Computers nicht überlasten können.

Die zweite Einschränkung - das Verbot der Verwendung der DLL beim Testen von Experten. DLL-Aufrufe sind auf Remote-Agenten aus Sicherheitsgründen absolut verboten. Auf der lokalen Agenten können DLL-Aufrufe nur mit der entsprechenden Erlaubnis "DLL-Import erlauben" in getesteten Experten erlaubt werden.



**Hinweis:** Bei der Verwendung der von Expert Advisors erhaltenen (Skripts, Indikatoren), die die DLL-Aufrufe zu erlauben erfordern, sollten Sie sich der Risiken bewusst sein, die Sie übernehmen falls Sie diese Option in den Einstellungen des Terminals erlauben. Unabhängig davon, wie der Expert-Advisor verwendet wird - für das Testen oder für den Lauf auf dem Chart.

## Zugeordnete Konstanten

für jedes ausführbare mql5-Programm wird eine Reihe von vorbestimmten Variablen unterstützt, die den Zustand des laufenden Preischarts widerspiegeln, wenn das Programm startet - Expert, Script oder Benutzeranzeiger.

Werte der vorbestimmten Variablen werden vom Kundenterminal vor dem Start des mql5-Programms eingestellt. Vorbestimmte Variablen sind konstant und können nicht aus dem mql5-Programm verändert werden. Die Ausnahme ist die Variable `_LastError`, die durch die Funktion [ResetLastError](#) ausgenullt werden kann.

| Variable                      | Wert                                                         |
|-------------------------------|--------------------------------------------------------------|
| <a href="#">_Digits</a>       | Anzahl von Dezimalzeichen nach dem Komma                     |
| <a href="#">_Point</a>        | Punktgrösse des laufenden Symbols in der Quotation Währung   |
| <a href="#">_LastError</a>    | Kode des letzten Fehlers                                     |
| <a href="#">_Period</a>       | Timeframe des laufenden Charts                               |
| <a href="#">_RandomSeed</a>   | Der aktuelle Zustand des Generators von Pseudo-Zufallszahlen |
| <a href="#">_StopFlag</a>     | Programmstop Flagge                                          |
| <a href="#">_Symbol</a>       | Symbolname des laufenden Charts                              |
| <a href="#">_UninitReason</a> | Kode des Grundes der Programmdeinitialisierung               |

Bibliotheken verwenden die Variablen der Programme, die sie aufgerufen hat.

## int \_Digits

Die Variable \_Digits bewahrt die Anzahl von Ziffern nach Dezimalpunkt auf, die die Preisgenauigkeit des Symbols des laufenden Charts definiert.

Man kann auch die Funktion [Digits\(\)](#) verwenden.

## double \_Point

Die Variable \_Point bewahrt die Punktgroesse des laufenden Symbols in Quotation Waehrung auf.

Man kann auch die Funktion [Point\(\)](#) verwenden.

## int \_LastError

Die Variable \_LastError bewahrt den Kode des letzten [Fehlers](#) auf, der während der Ausführung vom mql5-Programm entstand. Durch die Funktion [ResetLastError\(\)](#) kann der Wert auf Null gestellt werden.

Um Fehlerkode zu bekommen kann auch die Funktion [GetLastError\(\)](#) verwendet werden.



## int \_Period

Die Variable \_Period bewahrt Timeframe des laufenden Charts auf.

Man kann auch die Funktion [Period\(\)](#) verwenden.

Sehen Sie auch

[PeriodSeconds](#), [Chartperioden](#), [Datum und Zeit](#), [Objektsichtbarkeit](#)

## RandomSeed

Eine Variable für Speicherung des aktuellen Zustandes des Generators von Pseudo-Zufallszahlen RandomSeed ändert seinen Wert beim Aufruf von MathRand(). Um den richtigen Anfangszustand zu setzen verwenden Sie MathSrand().

Die Zufallszahl  $x$ , die durch die Funktion `MathRand()` erhalten wird, wird bei jedem Aufruf wie folgt berechnet:

```
x= _RandomSeed*214013+2531011;  
_RandomSeed=x;  
x= (x>>16) &0x7FFF;
```

Sehen Sie auch

[MathRand\(\)](#), [MathSrand\(\)](#), [Integer-Typen](#)

## bool \_StopFlag

Die Variable \_StopFlag bewahrt Programmstop Flagge des mql5-Programms. Wenn das Kundenterminal das Programm zu stoppen versucht, wird in diese Variable auf true eingestellt.

Für Prüfung des Status der Flagge \_StopFlag kann auch die Funktion [IsStopped\(\)](#) verwendet werden.

## string \_Symbol

In der Variable \_Symbol wird der Symbolname des laufenden Charts aufbewahren.

Man kann auch die Funktion [Symbol\(\)](#) verwenden.

## int \_UninitReason

In der Variable `_UninitReason` wird der Kode des [Deinitialisierungsgrundes](#) des Programms aufbewahren.

Gewöhnlich erhält man den Kode des Initialisierungsgrundes des Programms durch die Funktion [UninitializeReason\(\)](#).

## Allgemeine Funktionen

Funktionen allgemeiner Bedeutung, die keiner Sondergruppe angehören.

| Funktion                          | Massnahme                                                                                                                                                                 |
|-----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">Alert</a>             | Gibt Meldungen im selbstständigen Fenster aus                                                                                                                             |
| <a href="#">CheckPointer</a>      | Gibt Typ des Objektes zurück                                                                                                                                              |
| <a href="#">Comment</a>           | Gibt die Meldung in den oberen linken Winkel des Preischarts aus                                                                                                          |
| <a href="#">CryptEncode</a>       | Преобразует данные массива-источника в массив-приемник указанным методом                                                                                                  |
| <a href="#">CryptDecode</a>       | Производит обратное преобразование данных массива                                                                                                                         |
| <a href="#">DebugBreak</a>        | Programmhaltepunkt beim Debugging                                                                                                                                         |
| <a href="#">ExpertRemove</a>      | Stoppt die Expertarbeit und lädt Experten aus dem Chart aus                                                                                                               |
| <a href="#">GetPointer</a>        | Gibt <a href="#">Anzeiger</a> des Objektes zurück.                                                                                                                        |
| <a href="#">GetTickCount</a>      | Gibt die Anzahl der Millisekunden, die seit Systemstart vorbei sind                                                                                                       |
| <a href="#">MessageBox</a>        | Erzeugt und exponiert Meldungsfenster und auch verwaltet es                                                                                                               |
| <a href="#">PeriodSeconds</a>     | Gibt Sekundenanzahl in der Periode zurück                                                                                                                                 |
| <a href="#">PlaySound</a>         | Spielt Audiodatei ab                                                                                                                                                      |
| <a href="#">Print</a>             | Gibt die Meldung ins Magazin aus                                                                                                                                          |
| <a href="#">PrintFormat</a>       | Formatiert und druckt Schriftsatz und Werte in die Log-Datei aus entsprechend dem vorgegebenen Format                                                                     |
| <a href="#">ResetLastError</a>    | Stellt den Wert der vorbestimmten Variable <a href="#">_LastError</a> auf Null                                                                                            |
| <a href="#">ResourceCreate</a>    | Erstellt eine Ressource eines Bildes basierend auf dem Datensatz                                                                                                          |
| <a href="#">ResourceFree</a>      | Löscht <a href="#">dynamisch erstellte Ressource</a> (freit den Speicher)                                                                                                 |
| <a href="#">ResourceReadImage</a> | Liest Daten einer Grafik-Ressource, die <a href="#">von der Funktion ResourceCreate() erstellt war</a> oder <a href="#">in EX5-Datei beim Kompilieren gespeichert war</a> |
| <a href="#">ResourceSave</a>      | Speichert die Ressource in eine angegebene Datei                                                                                                                          |
| <a href="#">SendFTP</a>           | Sendet Datei an die Adresse, angegeben im                                                                                                                                 |

|                                         |                                                                                                                                                                   |
|-----------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                         | Fenster der Einstellungen im Registerblatt "Publikation"                                                                                                          |
| <a href="#"><u>SendMail</u></a>         | Sendet E-mail an die Adresse, angegeben im Fenster der Einstellungen im Registerblatt "Post"                                                                      |
| <a href="#"><u>SendNotification</u></a> | Sendet Push-Benachrichtigungen zu mobilen Terminals, deren MetaQuotes-IDs in das Optionen-Fenster auf der Registerkarte "Benachrichtigungen" angegeben sind       |
| <a href="#"><u>Sleep</u></a>            | Hemmt die Ausführung des laufenden Experten oder Script für bestimmte Zeit                                                                                        |
| <a href="#"><u>TerminalClose</u></a>    | Sendet dem Terminal Adressbefehl, Arbeit zu beenden                                                                                                               |
| <a href="#"><u>TesterStatistics</u></a> | Es gibt den Wert der statistischen Parameter, der nach Testergebnisse berechnet ist, zurück                                                                       |
| <a href="#"><u>WebRequest</u></a>       | Отправляет HTTP-запрос на указанный сервер                                                                                                                        |
| <a href="#"><u>ZeroMemory</u></a>       | Nullt die Variable aus, die durch Referenz übertragen wird. Typ der Variable kann verschieden sein, Ausnahmen bilden nur Klassen und Strukturen mit Konstruktoren |

## Alert

Wiederspiegelt Dialogfenster mit Benutzerdaten.

```
void Alert(  
    argument,      // der erste Wert  
    ...           // andere Werte  
);
```

### Parameter

*argument*

[in] Jede Werte, getrennt durch Kommas. Für Trennen die Informationsausgabe in mehrere Zeilen kann das Zeilenvorschubszeichen "\n" oder "\r\n" verwendet werden. Parameterzahl kann nicht mehr als 64 sein.

### Rückgabewert

Keinen Rückgabewert

### Bemerkung

Felder können nicht in die Funktion Alert() übertragen werden. Felder müssen elementenweise ausgegeben werden. Daten des Typs double werden mit 8 Dezimalzeichen, Daten des Typs float mit 5 Dezimalzeichen ausgegeben. für Ausgabe der reellen Zahlen mit anderer Genauigkeit oder im anderen Format muss die Funktion [DoubleToString\(\)](#) verwendet werden.

Daten des Typs bool werden als Zeilen "true" oder "false" ausgegeben. Daten werden als YYYY.MM.DD HH:MM:SS ausgegeben. Für Ausgabe des Datums im anderen Format muss die Funktion [TimeToString\(\)](#) verwendet werden. Daten des Typs color werden als R,G,B Zeile oder als Farbname ausgegeben, wenn es diese Farbe im Farbensatz gibt.

Beim Anwenden im [Strategie-Tester](#) wird die Alert()-Funktion nicht ausgeführt.



## CheckPointer

Gibt Typ des [Anzeigers](#) des Objektes zurück.

```
ENUM_POINTER_TYPE CheckPointer(
    object* anyobject    // Objektanzeiger
);
```

### Parameter

*anyobject*  
[in] Objektanzeiger.

### Rückgabewert

Gibt Wert aus der Aufzählung [ENUM\\_POINTER\\_TYPE](#) zurück.

### Bemerkung

Versuch der Anwendung eines unkorrekten Anzeiger führt zum [kritischen Brechen](#) des Programms. Darum ist es notwendig, die Funktion `CheckPointer` zu verwenden vor dem Gebrauch des Anzeigers. Anzeiger kann in folgenden Fällen unkorrekt sein:

- Anzeiger ist [NULL](#) gleich;
- wenn Objekt durch die Anweisung [delete](#) entfernt wurde.

Die vorliegende Funktion kann als Prüfung des Anzeigers auf Gültigkeit verwendet werden. Nicht-Null Wert garantiert, dass der Anzeiger für Zugang verwendet werden kann. .

### Beispiel:

```
//+-----+
//| Entfernung der Liste durch Entfernung der Elementen |
//+-----+
void CMyList::Destroy()
{
    //--- Dienstanzeiger für Arbeit im Zyklus
    CItem* item;
    //--- gehen wir durch den Zyklus und versuchen wir, dynamische Anzeiger zu entfernen
    while(CheckPointer(m_items) != POINTER_INVALID)
    {
        item=m_items;
        m_items=m_items.Next();
        if(CheckPointer(item) == POINTER_DYNAMIC)
        {
            Print("Dynamic object ",item.Identifier()," to be deleted");
            delete (item);
        }
        else Print("Non-dynamic object ",item.Identifier()," cannot be deleted");
    }
    //---
}
```

### Sehen Sie auch

[Objektanzeiger](#), [Prüfung der Objektanzeiger](#), [Anweisung der Objektentfernung delete](#)

## Comment

Gibt Kommentar, definiert vom Benutzer, in den oberen Chartwinkel aus.

```
void Comment(
    argument,      // der erste Wert
    ...           // andere Werte
);
```

### Parameter

...

[in] Jede Werte, getrennt durch Kommas. Für Trennen der Informationsausgabe in mehrere Zeichen kann das Zeilenvorschubszeichen "\n" oder "\r\n" verwendet werden. Parameterzahl kann nicht mehr als 64 sein. Die allgemeine Laenge der ausgegebenen Meldung (einschliesslich die nicht dargestellten Dienstsymbole) kann nicht mehr als 2045 sein (andere Symbole werden bei der Ausgabe entfernt werden).

### Rückgabewert

Keinen Rückgabewert

### Bemerkung

Felder können nicht in die Funktion Comment() übertragen werden. Felder muessen elementenweise gedruckt werden.

Daten des Typs double werden mit Genauigkeit bis 16 Dezimalzeichen ausgegeben entweder im traditionellen oder im wissenschaftlichen Format - abhängig davon, welches Dateneingabeformat kompakter wird. Daten des Typs float werden mit 5 Dezimalzeichen ausgegeben. Für Ausgabe der reellen Zahlen mit anderer Genauigkeit oder im vorbestimmten Format muss die Funktion [DoubleToString\(\)](#) verwendet werden.

Daten des Typs bool werden als Zeilen "true" oder "false" ausgegeben. Daten werden als YYYY.MM.DD HH:MI:SS ausgegeben. Für Ausgabe des Datums im anderen Format muss die Funktion [TimeToString\(\)](#) verwendet werden. Daten des Typs color werden entweder als R,G,B Zeile oder als Farbename, wenn es diese Farbe im Farbensatz gibt.

### Beispiel:

```
void OnTick()
{
    //---
    double Ask,Bid;
    int Spread;
    Ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
    Bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
    Spread=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD);
    //--- Geben wir Werte in drei Zeilen aus
    Comment(StringFormat("Ausgabe der Preise\nAsk = %G\nBid = %G\nSpread = %d",Ask,Bid,
    ;}
```

### Sehen Sie auch

[ChartSetString](#), [ChartGetString](#)

## CryptEncode

Преобразует данные массива-источника в массив-приемник указанным методом.

```
int CryptEncode(
    ENUM_CRYPT_METHOD method,      // метод преобразования
    const uchar& data[],           // массив-источник
    const uchar& key[],            // ключ шифрования
    const uchar& result[]          // массив-приемник
);
```

### Параметры

*method*

[in] Метод преобразования. Может быть одним из значений перечисления [ENUM\\_CRYPT\\_METHOD](#).

*data[]*

[in] Массив-источник.

*key[]*

[in] Ключ шифрования.

*result[]*

[out] Массив-приемник.

### Возвращаемое значение

Количество байт в массиве-приемнике или 0 в случае ошибки. Чтобы получить дополнительную информацию об [ошибке](#), необходимо вызвать функцию [GetLastError\(\)](#).

### Пример:

```
//+-----+
//| ArrayToHex |
//+-----+
string ArrayToHex(uchar &arr[],int count=-1)
{
    string res="";
    //--- проверка размера
    if(count<0 || count>ArraySize(arr))
        count=ArraySize(arr);
    //--- преобразование в шестнадцатичную строку
    for(int i=0; i<count; i++)
        res+=StringFormat("%.2X",arr[i]);
    //---
    return(res);
}
//+-----+
//| Script program start function |
//+-----+
void OnStart()
```

```

{
    string text="The quick brown fox jumps over the lazy dog";
    string keystr="ABCDEFGH";
    uchar src[],dst[],key[];
    ///--- подготовка ключа шифрования
    StringToArray(keystr,key);
    ///--- подготовка исходного массива src[]
    StringToArray(text,src);
    ///--- вывод исходных данных
    PrintFormat("Initial data: size=%d, string='%s'",ArraySize(src),CharArrayToString(src));
    ///--- шифрование массива src[] методом DES с 56-битным ключом key[]
    int res=CryptEncode(CRYPT_DES,src,key,dst);
    ///--- проверка результата шифрования
    if(res>0)
    {
        ///--- вывод зашифрованных данных
        PrintFormat("Encoded data: size=%d %s",res,ArrayToHex(dst));
        ///--- расшифровка данных массива dst[] методом DES с 56-битным ключом key[]
        res=CryptDecode(CRYPT_DES,dst,key,src);
        ///--- проверка результата
        if(res>0)
        {
            ///--- вывод дешифрованных данных
            PrintFormat("Decoded data: size=%d, string='%s'",ArraySize(src),CharArrayToString(src));
        }
        else
            Print("Ошибка в CryptDecode. Код ошибки=",GetLastError());
    }
    else
        Print("Ошибка в CryptEncode. Код ошибки=",GetLastError());
}

```

Смотри также

[Операции с массивами](#), [CryptDecode\(\)](#)

## CryptDecode

Производит обратное преобразование данных массива, полученного при помощи функции [CryptEncode\(\)](#).

```
int CryptDecode(  
    ENUM_CRYPT_METHOD  method,      // метод преобразования  
    const uchar&        data[],      // массив-источник  
    const uchar&        key[],       // ключ шифрования  
    const uchar&        result[]     // массив-приемник  
);
```

### Параметры

*method*

[in] Метод преобразования. Может быть одним из значений перечисления [ENUM\\_CRYPT\\_METHOD](#).

*data[]*

[in] Массив-источник.

*key[]*

[in] Ключ шифрования.

*result[]*

[out] Массив-приемник.

### Возвращаемое значение

Количество байт в массиве-приемнике или 0 в случае ошибки. Чтобы получить дополнительную информацию об [ошибке](#), необходимо вызвать функцию [GetLastError\(\)](#).

### Смотри также

[Операции с массивами](#), [CryptEncode\(\)](#)

## DebugBreak

Programmhaltepunkt beim Debugging .

```
void DebugBreak();
```

### Rückgabewert

Keinen Rückgabewert.

### Bemerkung

Die Ausführung des mql5-Programms wird es dann abgebrochen, wenn das Programm im Debugging Modus gestartet wird. Die Funktion kann verwendet werden, um Werte der Variablen durchzusehen oder für weitere gestufte Ausführung.

## ExpertRemove

Stoppt die Arbeit des [Experten](#) und laedt ihn aus dem Chart aus.

```
void ExpertRemove();
```

### Rückgabewert

Keinen Rückgabewert.

### Bemerkung

Expert wird nicht sofort beim Aufuf der Funktion ExpertRemove() gestoppt, es wird erst die Flagge für Brechen der Expertarbeit eingestellt. D.h. jedes folgendes Ereignis wird vom Experten schon nicht verarbeitet, [OnDeinit\(\)](#) wird aufgerufen werden und Expert wird ausgeladen werden und vom Chart entfernt werden.

### Beispiel:

```
//+-----+
//|                                     Test_ExpertRemove.mq5 |
//|                                     Copyright 2009, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
input int ticks_to_close=20; // Tickzahl vor dem Expertausladen
//+-----+
//| Expert deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
//---
    Print(TimeCurrent(), ": ", __FUNCTION__, " reason code = ", reason);
//--- "clear" comment
    Comment("");
//---
}
//+-----+
//| Expert tick function |
//+-----+
void OnTick()
{
    static int tick_counter=0;
//---
    tick_counter++;
    Comment("\Vor dem Expertausladen ", __FILE__, " bleiben",
        (ticks_to_close-tick_counter), "Ticks");
//--- bis
    if(tick_counter>=ticks_to_close)
    {
        ExpertRemove();
        Print(TimeCurrent(), ": ", __FUNCTION__, " Expert wird ausgeladen werden");
    }
    Print("tick_counter = ", tick_counter);
//---
}
//+-----+
```



Sehen Sie auch

[Programmausführung](#), [Ereignisse des Kundenterminals](#)

## GetPointer

Gibt [Anzeiger](#) des Objektes zurück.

```
void* GetPointer(  
    any_class anyobject    // Objekt jeder Klasse  
);
```

### Parameter

*anyobject*

[in] Objekt jeder Klasse.

### Rückgabewert

Gibt Objektanzeiger zurück.

### Bemerkung

Nur Klassenobjekte haben Anzeiger. Instanzen der [Strukturen](#) und Variablen einfacher Typen haben keine Anzeiger. Klassenobjekt, das nicht durch die Anweisung new(), sondern ZB. automatisch im Feld der Objekte erzeugt wurde, hat trotzdem einen Anzeiger. Dieser Anzeiger wird aber des automatischen Typs POINTER\_AUTOMATIC sein, darum kann die Anweisung [delete\(\)](#) nicht angewendet werden. Sonst unterscheidet sich die Anweisung dieses Typs nicht von den Anzeigern des Typs [POINTER\\_AUTOMATIC](#).

Da Variablen des Typs Strukturen und einfacher Typen keine Anzeiger haben, ist es verboten, die Funktion GetPointer() für sie anzuwenden. Man darf auch nicht Anzeiger als Argument der Funktion zu übertragen. In allen genannten Fällen meldet der Compiler den Fehler.

Versuch der Anwendung zum unkorrekten Anzeiger führt zur [kritischen Beendigung](#) des Programms. Darum besteht es die Notwendigkeit, die Funktion [CheckPointer\(\)](#) vor dem Gebrauch des Anzeigers zu verwenden. Anzeiger kann in folgenden Fällen unkorrekt sein:

- Anzeiger ist [NULL](#) gleich;
- wenn Objekt durch die Anweisung [delete](#) entfernt wurde.

Diese Funktion kann als Prüfung des Anzeigers auf Validität verwendet werden. Nicht-Null Wert garantiert, dass Anzeiger für Zugang verwendet werden kann.

### Beispiel:

```

//+-----+
//|                                     Check_GetPointer.mq5 |
//|                                     Copyright 2009, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

//+-----+
//| Klasse, implementierte Element der Liste |
//+-----+
class CItem
{
    int          m_id;
    string       m_comment;
    CItem*       m_next;
public:
    CItem() { m_id=0; m_comment=NULL; m_next=NULL; }
    ~CItem() { Print("Destructor of ",m_id,
                    (CheckPointer(GetPointer(this))==POINTER_DYNAMIC)
                    "dynamic":"non-dynamic"); }

    void Initialize(int id,string comm) { m_id=id; m_comment=comm; }
    void PrintMe() { Print(__FUNCTION__,":",m_id,m_comment); }
    int Identifier() { return(m_id); }
    CItem* Next() {return(m_next); }
    void Next(CItem *item) { m_next=item; }
};

//+-----+
//| Die einfachste Klasse der Liste |
//+-----+
class CMyList
{
    CItem*       m_items;
public:
    CMyList() { m_items=NULL; }
    ~CMyList() { Destroy(); }

    bool InsertToBegin(CItem* item);
    void Destroy();
};

//+-----+
//| Einfuegung des Elementes der Liste am Anfang |
//+-----+
bool CMyList::InsertToBegin(CItem* item)
{
    if(CheckPointer(item)==POINTER_INVALID) return(false);
//---
    item.Next(m_items);
    m_items=item;
//---
    return(true);
}

//+-----+
//| Entfernung der Liste durch Entfernung der Elementen |
//+-----+
void CMyList::Destroy()
{
    //--- Dienstanzeiger für Arbeit im Zyklus
    CItem* item;
    //--- gehen wir durch den Zyklus und versuchen wir, dynamische Anzeiger zu entfernen
    while(CheckPointer(m_items)!=POINTER_INVALID)

```

```

    {
        item=m_items;
        m_items=m_items.Next();
        if(CheckPointer(item)==POINTER_DYNAMIC)
        {
            Print("Dynamyc object ",item.Identifier()," to be deleted");
            delete (item);
        }
        else Print("Non-dynamic object ",item.Identifier()," cannot be deleted");
    }
}
//---
}
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    CMyList list;
    CItem  items[10];
    CItem* item;
    //--- erzeugen wir und fuegen in die Liste das dynamische Anzeiger des Objektes hinzu
    item=new CItem;
    if(item!=NULL)
    {
        item.Initialize(100,"dynamic");
        item.PrintMe();
        list.InsertToBegin(item);
    }
    //--- fuegen wir automatische Anzeiger in die Liste hinzu
    for(int i=0; i<10; i++)
    {
        items[i].Initialize(i,"automatic");
        items[i].PrintMe();
        item=GetPointer(items[i]);
        if(CheckPointer(item)!=POINTER_INVALID)
            list.InsertToBegin(item);
    }
    //--- fuegen wir noch einen dynamischen Anzeiger des Objektes am Anfang der Liste hinzu
    item=new CItem;
    if(item!=NULL)
    {
        item.Initialize(200,"dynamic");
        item.PrintMe();
        list.InsertToBegin(item);
    }
    //--- entfernen wir Elementen der Liste
    list.Destroy();
    //--- Alle Elementen der Liste werden entfernt werden,
    //--- sieh im Terminal Registerblatt Experts
}

```

### Sehen Sie auch

[Objektanzeiger](#), [Pruefung des Objektanzeigers](#), [Anweisung der Objektentfernung delete](#)

## GetTickCount

Funktion GetTickCount() gibt die Zahl der Millisekunden, die seit Systemstart vorbei sind.

```
uint GetTickCount();
```

### Rückgabewert

Wert des Typs uint.

### Bemerkung

Counter ist durch Auflösungsvermögen des Systemtimers begrenzt. Da die Zeit als das zeichenlose Ganze aufbewahrt wird, wird er jede 49.7 Tage bei ununterbrochener Computerarbeit überfüllt.

### Beispiel:

```
#define MAX_SIZE 40
//+-----+
//| Skript für Messung der Rechenzeit von 40 Fibonacci-Zahlen |
//+-----+
void OnStart()
{
//--- Merken wir uns den Initialwert
    uint start=GetTickCount();
//--- Variable für Erhaltung der nächsten Zahl in der Fibonacci-Reihe
    long fib=0;
//--- Zyklus, in dem wir eine bestimmte Anzahl von Zahlen in der Fibonacci-Reihe berechnen
    for(int i=0;i<MAX_SIZE;i++) fib=TestFibo(i);
//--- Verstrichene Zeit in Millisekunden erhalten
    uint time=GetTickCount()-start;
//--- Eine Meldung für Journal "Experten"
    PrintFormat("Berechnung der %d ersten Fibonacci-Zahlen dauerte %d ms",MAX_SIZE,time);
//--- Das Skript ist fertig
    return;
};
//+-----+
//| Funktion für Erhaltung der Fibonacci-Zahle durch ihr Folgennummer |
//+-----+
long TestFibo(long n)
{
//--- Das erste Glied der Fibonacci-Reihe
    if(n<2) return(1);
//--- Alle nachfolgenden Glieder sind nach der Formel berechnet
    return(TestFibo(n-2)+TestFibo(n-1));
};
```

### Sehen Sie auch

[Datum und Zeit](#)

## MessageBox

Erzeugt und widerspiegelt das Meldungsfenster und verwaltet es. Meldungsfenster enthält Meldung und Kopf, jede Kombination der vorbestimmten Zeichen und Befehlsschaltflächen.

```
int MessageBox(  
    string text,           // Meldungstext  
    string caption=NULL,  // Meldungskopf  
    int flags=0           // definiert Schaltflächensatz im Fenster  
);
```

### Parameter

*text*

[in] Text mit der Meldung für Ausgabe.

*caption=NULL*

[in] Freier Text für Ausgabe im Kopf des Meldungsfensters. Wenn dieser Parameter leer ist, wird im Fensterkopf Expertenname wiedergespiegelt. .

*flags=0*

[in] Freie [Flaggen](#), die die Art und Verhalten des Dialogfensters bestimmen. Flaggen können die Kombination der Sondergruppe der Flaggen sein.

### Rückgabewert

Wenn die Funktion erfolgreich ausgeführt wird, ist der Rückgabewert einer der Werte der Rückgabekodes [MessageBox\(\)](#).

### Bemerkung

Funktion kann nicht aus Benutzerindikatoren aufgerufen werden, denn Anzeiger werden im Thread ausgeführt und müssen den Thread nicht hemmen.

Beim Anwenden im [Strategie-Tester](#) wird die MessageBox()-Funktion nicht ausgeführt.

## PeriodSeconds

Gibt die Zahl der Sekunden in der Periode zurück.

```
int PeriodSeconds(  
    ENUM_TIMEFRAMES period=PERIOD_CURRENT // Chartperiode  
);
```

### Parameter

*period=PERIOD\_CURRENT*

[in] Wert der Chartperiode aus der Aufzählung [ENUM\\_TIMEFRAMES](#). Wenn der Parameter nicht angegeben wird, wird die Sekundenzahl des Charts der laufenden Periode, wo das Programm gestartet ist, zurückgegeben.

### Rückgabewert

Zahl der Sekunden in der angegebenen Periode.

### Sehen Sie auch

[\\_Period](#), [Chartperioden](#), [Datum und Zeit](#), [Objektsichbarkeit](#)

## PlaySound

Spielt Audiodatei ab.

```
bool PlaySound(  
    string filename    // Dateiname  
);
```

### Parameter

*filename*

[in] Pfad zur Audiodatei. Wenn filename=NULL ist, wird die Audiowiedergabe gestoppt.

### Rückgabewert

true - wenn Audiodatei nicht gefunden wird, ansonsten false.

### Bemerkung

Datei muss sich im Katalog Katalog\_des Terminals\Sounds oder in seinem Subkatalog befinden. Es werden nur Audiodateien im Format WAV abgespielt.

Aufruf von Playsound() mit einem Parameter NULL stoppt Audiowiedergabe.

Beim Anwenden im [Strategie-Tester](#) wird die PlaySound()-Funktion nicht ausgeführt.

### Sehen Sie auch

[Ressourcen](#)



## Print

Druckt eine Nachricht im Expertenjournal. Parameter können verschiedener Typs sein.

```
void Print(  
    argument,      // der erste Wert  
    ...            // folgende Werte  
);
```

### Parameter

...

[in] Jede Werte, getrennt durch Kommas. Parameterzahl kann nicht mehr als 64 sein.

### Bemerkung

Parameter können nicht in die Funktion Print() übertragen werden. Felder müssen elementenweise gedruckt werden.

Daten des Typs double werden mit 16 Dezimalzeichen ausgegeben., dabei können Daten im traditionellen oder im wissenschaftlichen Format ausgegeben werden - abhängig davon, welches Ausgabeformat am kompaktesten ist. Daten des Typs float werden mit 5 Dezimalzeichen ausgegeben. Für Ausgabe der reellen Zahlen mit anderer Genauigkeit oder im vorbestimmten Format muss die Funktion [PrintFormat\(\)](#) verwendet werden.

Daten des Typs bool werden als Zeilen des Typs "true" oder "false" ausgegeben. Daten werden als YYYY.MM.DD HH:MI:SS ausgegeben . Für Ausgabe des Datums im anderen Format muss die Funktion [TimeToString\(\)](#) verwendet werden. Daten des Typs color werden entweder als die Zeile R,G,B oder als Farbename ausgegeben, wenn diese Farbe im Farbenbestand gibt.

Im [Strategie-Tester](#) im Optimierungsmodus wird die Print()-Funktion nicht ausgeführt.

### Beispiel:

```

void OnStart()
{
//--- Ausgeben DBL_MAX mit Print(), ist dies gleichbedeutend PrintFormat(%.16G,DBL_MAX)
Print("---- Wie DBL_MAX sieht aus ----");
Print("Print(DBL_MAX)=",DBL_MAX);
//--- Jetzt geben wir DBL_MAX mit PrintFormat() aus
PrintFormat("PrintFormat(%.16G,DBL_MAX)=%.16G",DBL_MAX);
//--- Ausgabe in Expertenjournal
// Print(DBL_MAX)=1.797693134862316e+308
// PrintFormat(%.16G,DBL_MAX)=1.797693134862316E+308

//--- Sehen wir wie Typ float ausgegeben wird
float c=(float)M_PI; // Wir müssen explizit auf das Zieltype umwandeln
Print("c=",c, "      Pi=",M_PI, "      (float)M_PI=", (float)M_PI);
// c=3.14159      Pi=3.141592653589793      (float)M_PI=3.14159

//--- Wir zeigen, was während den arithmetischen Operationen mit echten passieren kann
double a=7,b=200;
Print("---- vor den arithmetischen Operationen");
Print("a=",a, "      b=",b);
Print("Print(DoubleToString(b,16))=",DoubleToString(b,16));
//--- Wir dividieren a durch b (7/200)
a=a/b;
//--- Wiederherstellung des Werts von b
b=7.0/a; // Es wird erwartet, dass b=7.0/(7.0/200.0)=>7.0/7.0*200.0=200 - aber es ist nicht
//--- Wir geben den neu berechneten Wert von b
Print("----- nach den arithmetischen Operationen");
Print("Print(b)=",b);
Print("Print(DoubleToString(b,16))=",DoubleToString(b,16));
//--- Ausgabe in Expertenjournal
// Print(b)=200.0
// Print(DoubleToString(b,16))=199.999999999999716 (Wir sehen dass b nicht gleich 200 ist)

//--- Erstellen wir einen sehr kleinen Wert epsilon=1E-013
double epsilon=1e-13;
Print("---- Erstellen wir einen sehr kleinen Wert");
Print("epsilon=",epsilon); // Erhalten epsilon=1E-013
//--- Nun b weniger Epsilon - geben den Wert in Expertenjournal aus
b=b-epsilon;
//--- Ausgeben durch zwei Methoden
Print("---- Nach Subtraktion von epsilon aus b");
Print("Print(b)=",b);
Print("Print(DoubleToString(b,16))=",DoubleToString(b,16));
//--- Ausgabe in Expertenjournal
// Print(b)=199.99999999999999 (Nun kann der Wert von b nach Subtraktion von epsilon nicht bis zu 200 gerundet werden)
// Print(DoubleToString(b,16))=199.999999999998578
// (Nun kann der Wert von b nach Subtraktion von epsilon nicht bis zu 200 gerundet werden)
}

```

Sehen Sie auch

[DoubleToString](#), [StringFormat](#)

# PrintFormat

Formatiert und druckt Symbolvorrat im Expertenjournal laut dem vorgegebenen Format.

```
void PrintFormat(
    string format_string,    // Formatzeile
    ...                     // Werte einfacher Typen
);
```

## Parameter

- format\_string*
- [in] Formatzeile besteht aus den normalen Symbolen und auch Formatspezifikation (wenn Argumente nach der Formatzeile stehen).
- ...
- [in] Jede Werte einfacher Typen, die durch Kommas getrennt werden. Die gesamte Anzahl der Parameter kann nicht mehr als 64 sein, einschliesslich die Formatzeile.

## Rückgabewert

Zeile.

## Bemerkung

Im [Strategie-Tester](#) im Optimierungsmodus wird die PrintFormat()-Funktion nicht ausgeführt.

Anzahl, Reihenfolge und Art der Parameter muss exakt der Zusammensetzung der Qualifikanten entsprechen, sonst das Druckergebnis ist undefiniert. Statt der Funktion PrintFormat() kann die Funktion [printf\(\)](#) verwendet werden.

Wenn nach der Zeile Parameter stehen, muss diese Zeile Formatspezifikationen enthalten, die Ausgabeformat dieser Parameter bestimmen. Formatspezifikation beginnt mit Symbol (%).

Formatzeile wird von links nach rechts gelesen. Wenn die erste Formatspezifikation auftritt (wenn es gibt), wird der Wert des ersten Parameters nach der Formatzeile umgewandelt und laut der vorgegebenen Spezifikation ausgegeben. Die zweite Spezifikation ruft Umwandlung und Ausgabe des zweiten Parameters auf usw bis zum Ende der Formatzeile.

Formatspezifikation sieht so aus:

**%[flags][width][.precision][{h | l | ll | l32 | l64}]type**

Jedes Feld der Formatspezifikation ist entweder ein einfaches Symbol oder eine Zahl, die eine normale Formatoption bezeichnet. Die einfachste Formatspezifikation enthält nur Prozentzeichen (%) und Symbol, das den [Typ des Eingabeparameters](#) (ZB %s) bestimmt. Wenn das Prozentzeichen in der Formatzeile ausgegeben werden muss, muss die Formatspezifikation %%. verwendet werden.

## flags

| Flagge    | Beschreibung              | Default-Verhalten          |
|-----------|---------------------------|----------------------------|
| - (minus) | Linksbuendige Ausrichtung | Rechtsbuendige Ausrichtung |

|          |                                                                                                                                                                                                                                                                 |                                                                                                                      |
|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------|
|          | innerhalb der vorgegebenen Breite                                                                                                                                                                                                                               |                                                                                                                      |
| + (plus) | Ausgabe des Zeichens + oder - für Werte der Zeichentypen                                                                                                                                                                                                        | Zeichen wird ausgegeben, nur wenn der Wert negativ ist.                                                              |
| 0 (Null) | Vor dem Ausgabewert werden Nullen innerhalb der vorgegebenen <a href="#">Breite</a> zugesetzt. Wenn die Flagge 0 mit dem ganzzahligen Format (i, u, x, X, o, d) angegeben wird und Spezifikation der Genauigkeit vorgegeben wird (ZB, %04.d), wird 0 ignoriert. | Nichts wird eingegeben                                                                                               |
| Space    | Vor dem Ausgabewert wird Space gestellt, wenn der Wert ein positiver Zeichenwert ist.                                                                                                                                                                           | Spaces werden nicht eingegeben                                                                                       |
| #        | Beim Joint-Verwenden mit o, x oder X wird vor dem Ausgabewert 0, 0x oder 0X zugesetzt.                                                                                                                                                                          | Nichts wird eingegeben                                                                                               |
|          | Beim Joint-Verwenden mit Format e, E, a oder A wird der Wert immer mit Dezimalpunkt ausgegeben.                                                                                                                                                                 | Dezimalpunkt wird erst dann ausgegeben, wenn es einen Nicht-Null Bruchanteil gibt.                                   |
|          | Beim Joint-Verwenden mit Format g oder G, bestimmt Flagge das Vorhandensein des Dezimalpunktes und beugt die Reduzierung der führenden Nullen vor.<br>Flagge # wird beim Joint-Verwenden mit Formaten c, d, i, u, s ignoriert.                                  | Dezimalpunkt wird erst dann ausgegeben, wenn es einen Nicht-Null Bruchanteil gibt. Führende Nullen werden reduziert. |

## width

Eine nichtnegative Dezimalzahl, die minimale Anzahl der Ausgabesymbole des formatierten Wertes vorgibt. Wenn die Anzahl der Ausgabesymbole weniger als angegebene Breite ist, wird die entsprechende Anzahl der Spaces links und rechts abhängig von Ausrichtung zugesetzt (Flagge -). Beim Vorhandensein der Flagge Null, wird vor dem Ausgabewert die entsprechende Anzahl von Nullen zugesetzt. Wenn die Anzahl der Ausgabesymbole mehr als die vorgegebene Breite ist, wird der Ausgabewert nie reduziert.

Wenn als Breite Sternchen (\*) angegeben wird, muss in der Liste der übertragenen Parameter der Wert des Typs int an der entsprechenden Stelle sein, der für Breiteangabe des Ausgabewertes verwendet wird.

## precision

Eine nichtnegative Dezimalzahl, die Genauigkeit der Ausgabe bestimmt - Anzahl der Dezimalzeichen. Zum Unterschied von Spezifikation der Länge kann Spezifikation der Genauigkeit einen Teil des Bruchwertes mit Aufrundung oder ohne Aufrundung reduzieren.

für verschiedene [Typen](#) (type) des Formats wird Spezifikation der Genauigkeit verschieden verwendet.

| Typen            | Beschreibung                                                                                                                                                                                                                                                                                                        | Default-Verhalten                                                                                                               |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------|
| a, A             | Spezifikation der Genauigkeit zeigt Anzahl der Dezimalzeichen                                                                                                                                                                                                                                                       | Default-Genauigkeit - 6.                                                                                                        |
| c, C             | nicht verwendet                                                                                                                                                                                                                                                                                                     |                                                                                                                                 |
| d, i, u, o, x, X | Zeigt minimale Zahl der ausgegebenen Ziffern. Wenn die Anzahl der Ziffern im vorgegebenen Parameter weniger als die angegebene Genauigkeit ist, wird der Ausgabewert links durch Nullen ergaenzt, Der Ausgabewert wird nicht reduziert, wenn die Anzahl der Ausgabeziffern mehr als die angegebene Genauigkeit ist. | Default-Genauigkeit - 1.                                                                                                        |
| e, E, f          | Gibt die Anzahl der Dezimalzeichen an, die letzte Ausgabeziffer wird ausgerundet.                                                                                                                                                                                                                                   | Default-Genauigkeit - 6. Wenn die Genauigkeit 0 angegeben wird oder der Bruchteil fehlt, wird der Dezimalpunkt nicht ausgegeben |
| g, G             | Gibt die maximale Anzahl der bedeutsamen Ziffern an                                                                                                                                                                                                                                                                 | Es wird 6 bedeutsame Ziffern ausgegeben.                                                                                        |
| s, S             | Gibt die Anzahl der Ausgabezeilensymbole an. Wenn die Zeilenlaenge mehr als der Wert der Genauigkeit ist, wird sie bei der Ausgabe reduziert.                                                                                                                                                                       | die ganze Zeile wird ausgegeben                                                                                                 |

Sehen Sie auch

[h](#) | [l](#) | [ll](#) | [I32](#) | [I64](#)

Spezifikation der Laengedimensionen, die als Parameter übertragen werden.

| Typ des Parameters | Das verwendete Praefix | Joint-Spezifikator des Typs |
|--------------------|------------------------|-----------------------------|
| int                | l (kleiner L)          | d, i, o, x, or X            |
| uint               | l (kleiner L)          | o, u, x, or X               |
| long               | ll (zwei kleine L)     | d, i, o, x, or X            |
| short              | h                      | d, i, o, x, or X            |
| ushort             | h                      | o, u, x, or X               |
| int                | l32                    | d, i, o, x, or X            |
| uint               | l32                    | o, u, x, or X               |
| long               | l64                    | d, i, o, x, or X            |
| ulong              | l64                    | o, u, x, or X               |

## type

Spezifikator des Typs ist ein einziges obligatorische Feld für formatierte Ausgabe.

| Symbol | Typ    | Ausgabeformat                                                                                                                                                                                    |
|--------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| c      | int    | Symbol des Typs short (Unicode)                                                                                                                                                                  |
| C      | int    | Symbol des Typs char (ANSI)                                                                                                                                                                      |
| d      | int    | Zeichendezimalganzzahl                                                                                                                                                                           |
| i      | int    | Zeichendezimalganzzahl                                                                                                                                                                           |
| o      | int    | Zeichenlose Oktalganzzahl                                                                                                                                                                        |
| u      | int    | Zeichenlose Dezimalganzzahl                                                                                                                                                                      |
| x      | int    | Zeichenlose hexadezimalganzzahl mit "abcdef" Verwendung                                                                                                                                          |
| X      | int    | Zeichenlose hexadezimalganzzahl mit "ABCDEF" Verwendung                                                                                                                                          |
| e      | double | Realwert in Format [ - ] d.dddd e [sign]ddd, wo d - eine Dezimalzahl, dddd - eine oder mehrere Dezimalzahlen, ddd - dreistellige Zahl, die Exponentengroesse bestimmt, Zeichen - plus oder minus |
| E      | double | Dem Format e, gleich unter                                                                                                                                                                       |

|   |        |                                                                                                                                                                                                                                                                        |
|---|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|   |        | Ausschluss, dass Exponentenzeichen mit einem Grossbuchstaben anfaengt (E statt e)                                                                                                                                                                                      |
| f | double | Realwert in Format [ - ] dddd.dddd, wo dddd - eine oder mehrere Dezimalziffern. Anzahl der ausgegebenen Dezimalzeichen hängt von der Größe der Zahl ab. Anzahl der Dezimalzeichen hängt von der erforderlichen Genauigkeit ab.                                         |
| g | double | Realwert, der im Format f oder e ausgegeben wird, abhängig davon, welches Format kompakter ist.                                                                                                                                                                        |
| G | double | Realwert der im Format f oder e ausgegeben wird, abhängig davon, welche Ausgabe kompakter ist.                                                                                                                                                                         |
| a | double | Realwert in Format [-] 0xh.hhhh p±dd, wo h.hhhh - mantissa in Form der hexadezimaler Ziffern mit Verwendung von "abcdef", dd - eine oder mehrere Ziffern einer Exponente. Anzahl der Dezimalzeichen wird durch <a href="#">Spezifikation der Genauigkeit</a> bestimmt. |
| A | double | Realwert in Format [-] 0xh.hhhh P±dd, wo h.hhhh - mantissa in Form von hexadezimaler Ziffern mit Verwendung von "ABCDEF", dd - eine oder mehrere Ziffern einer Exponente. Anzahl der Dezimalzeichen wird durch <a href="#">Spezifikation der Genauigkeit</a> bestimmt. |
| s | string | Zeilenausgabe                                                                                                                                                                                                                                                          |

Anstatt der Funktion PrintFormat() kann die Funktion `printf()` verwendet werden.

**Beispiel:**

```
PrintFormat("Anzeige DBL_MAX in eine kompakte wissenschaftliche Form: %e",DBL_MAX);  
printf("Anzeige DBL_MAX wie %.15e: %.15e",DBL_MAX);  
printf("Anzeige DBL_MAX wie %%15e: %15e",DBL_MAX);  
printf("Anzeige DBL_MAX wie %%15.10e: %15.10e",DBL_MAX);  
printf("Anzeige DBL_MAX wie %%15.10f: %15.10f",DBL_MAX);  
  
printf("Anzeige 10 wie %%f: %f",10);  
printf("Anzeige DBL_MAX wie %%d: %d",DBL_MAX);  
printf("Anzeige 10 wie %%e: %e",10);  
printf("Anzeige DBL_MAX wie %%e: %e",DBL_MAX);
```

Sehen Sie auch

[StringFormat](#), [DoubleToString](#), [Realtypen \(double, float\)](#)



## ResetLastError

Stellt den Wert der vorbestimmten Variable [\\_LastError](#) auf Null ein.

```
void ResetLastError();
```

### Rückgabewert

Keinen Rückgabewert.

### Bemerkung

Es muss erwähnt werden, dass die Funktion [GetLastError\(\)](#) nullt die Variable `_LastError` nicht aus. Gewöhnlich wird die Funktion `ResetLastError()` vor dem Aufruf der Funktion aufgerufen, nach der die Entstehung des [Fehlers](#) geprüft wird.

## ResourceCreate

Erstellt eine Ressource eines Bildes basierend auf dem Datensatz. Es gibt zwei Versionen der Funktion:  
**Erstellen einer Datei-basierte Ressource**

```
bool ResourceCreate(
    const string      resource_name,      // Ressourcename
    const string      path                // Ein relativer Pfad zur Datei
);
```

**Erstellen einer Ressource basierend auf einem Array von Pixeln**

```
bool ResourceCreate(
    const string      resource_name,      // Name der Ressource
    const uint&       data[],             // Datensatz im Form eines Arrays
    uint             img_width,          // Breite der erstellten Bildressource
    uint             img_height,         // Höhe der erstellten Bildressource
    uint             data_xoffset,        // Horizontales Offset der oberen linken Eck
    uint             data_yoffset,        // Vertikales Offset der oberen linken Ecke
    uint             data_width,          // Gesamtbreite des Bildes basierend auf der
    ENUM_COLOR_FORMAT color_format        // Farbe Verarbeitungsmethode
);
```

### Parameter

*resource\_name*

[in] Name der Ressource.

*data[] []*

[in] Eindimensionales oder zweidimensionales Array um ein vollständiges Bild zu erstellen.

*img\_width*

[in] Breite der rechteckigen Bereich des Bildes in Pixel um in der Ressource in der Form von Bildern zu speichern. Es kann nicht mehr als *data\_width* sein.

*img\_height*

[in] Höhe der rechteckigen Bereich des Bildes in Pixel um in der Ressource in der Form von Bildern zu speichern.

*data\_xoffset*

[in] Horizontales Offset der rechteckigen Bereich des Bildes in Pixel nach rechts.

*data\_yoffset*

[in] Vertikales Offset der rechteckigen Bereich des Bildes in Pixel nach unten.

*data\_width*

[in] Nur für eindimensionale Arrays erforderlich ist, und bedeutet die gesamte Breite des Bildes aus dem Datensatz. Wenn *data\_width=0*, wird gleich *img\_width* angenommen. Für zweidimensionale Arrays, wird dieser Parameter ignoriert, und es wird angenommen, dass er der zweite Dimension des Arrays *data[]* gleich ist.

*color\_format*

[in] Farbe Verarbeitungsmethode aus der Aufzählung [ENUM\\_COLOR\\_FORMAT](#).

### Rückgabewert

Gibt bei Erfolg true zurück, ansonsten false. Um Informationen über den Fehler zu erhalten, rufen Sie die Funktion [GetLastError\(\)](#). Mögliche Fehler:

- 4015 - ERR\_RESOURCE\_NAME\_DUPLICATED (die Namen der dynamischen und [statischen](#) Ressourcen sind gleich)
- 4016 - ERR\_RESOURCE\_NOT\_FOUND (Ressource nicht gefunden)
- 4017 - ERR\_RESOURCE\_UNSUPPORTED\_TYPE (Ressource-Typ wird nicht unterstützt)
- 4018 - ERR\_RESOURCE\_NAME\_IS\_TOO\_LONG (Name der Ressource ist zu lang).

#### Bemerkung

Wenn die zweite Version der Funktion aufgerufen wird, um die gleiche Ressource mit unterschiedlichen Parametern von Breite, Höhe und Offset zu erstellen, ist die neue Ressource nicht erstellt, sondern einfach eine vorhandene Ressource aktualisiert wird.

Die erste Version der Funktion erlaubt Ihnen, Bilder und Sounds aus Dateien zu laden, wird die zweite Version nur für die dynamische Erstellung von Bildern verwendet.

Die Bilder müssen im BMP-Format mit einer Farbtiefe von 24 oder 32 Bit sein, Sounds können nur im Format WAV sein. Die Größe der Ressource sollte nicht mehr als 16 Mb sein.

#### ENUM\_COLOR\_FORMAT

| Identifizier                | Beschreibung                                                                                                    |
|-----------------------------|-----------------------------------------------------------------------------------------------------------------|
| COLOR_FORMAT_XRGB_NOALPHA   | Bestandteil der Alpha-Kanal wird ignoriert                                                                      |
| COLOR_FORMAT_ARGB_RAW       | Die Farbkomponenten sind durch Terminal nicht verarbeitet (soll korrekt durch den Benutzer spezifiziert werden) |
| COLOR_FORMAT_ARGB_NORMALIZE | Die Farbkomponenten sind durch Terminal verarbeitet                                                             |

#### Sehen Sie auch

[Ressourcen](#), [ObjectCreate\(\)](#), [ObjectSetString\(\)](#), [OBJPROP\\_BMPFILE](#)

## ResourceFree

Löscht dynamisch erstellte Ressource (freit den Speicher).

```
bool ResourceFree(  
    const string resource_name    // Ressource-Name  
);
```

### Optionen

*resource\_name*

[in] Name der Ressource, muss beginnen mit "::".

### Rückgabewert

true - wenn erfolgreich, andernfalls false. Um Informationen über den Fehler zu erhalten, rufen Sie GetLastError() an.

### Hinweis

Funktion ResourceFree() erlaubt es dem Entwickler von MQL5-Programm Speicherbedarf beim Arbeiten mit Ressourcen zu kontrollieren. Grafische Objekte, die an aus dem Speicher gelöschten Ressource gebunden sind, werden richtig nach ihrer Löschung angezeigt werden. Aber die neu erstellte graphische Objekte (OBJ\_BITMAP und OBJ\_BITMAP\_LABEL) können nicht die gelöschte Ressource nutzen.

Die Funktion löscht nur die dynamische Ressourcen, die durch das Programm erstellt werden.

### Sehen Sie auch

Ressourcen, ObjectCreate(), PlaySound(), ObjectSetString(), OBJPROP\_BMPFILE

## ResourceReadImage

Liest Daten einer Grafik-Ressource, die [von der Funktion ResourceCreate\(\) erstellt war](#) oder [in EX5-Datei beim Kompilieren gespeichert war](#).

```
bool ResourceReadImage (
    const string      resource_name,      // Name der Grafik-Ressource zum Lesen
    uint&             data[],             // Array, um die Daten aus der Ressource zu
    uint&             width,             // Um die Breite des Bildes in der Ressource
    uint&             height,           // Um die Bildhöhe in der Ressource zu erhalten
);
```

### Optionen

*resource\_name*

[in] Name der grafischen Ressource, die das Bild enthält. Um die eigenen Ressourcen zu zugreifen, der Name wird in der Kurzform von "::resourcenname" angegeben. Wenn Sie die Ressource aus der kompilierten EX5-Datei laden müssen, muss der volle Name mit dem relativen Pfad zu dem Ordner MQL5, den Dateinamen und Ressource-Name sein - "path\\filename.ex5::resourcenname".

*data[] []*

[in] Ein eindimensionales oder zweidimensionales Array in dem die Daten eines graphischen Ressource erhalten werden.

*img\_width*

[out] Bildbreite der Ressource in Pixels .

*img\_height*

[out] Bildhöhe der Ressource in Pixels .

### Rückgabewert

true - wenn erfolgreich, andernfalls false. Um Informationen über den Fehler zu erhalten, rufen Sie [GetLastError\(\)](#) an.

### Hinweis

Wenn auf der Grundlage des Arrays *data[]* brauchen Sie weiter [eine Grafik-Ressource](#) zu erstellen, sollten Sie [Farbe Format](#) COLOR\_FORMAT\_ARGB\_NORMALIZE oder COLOR\_FORMAT\_XRGB\_NOALPHA benutzen.

Wenn das Array *data[]* zweidimensional ist, und die zweite Dimension ist kleiner als die X(width) der grafischen Ressource, gibt die Funktion ResourceReadImage() false zurück und das Lesen wird nicht vorgenommen. Aber in diesem Fall, wenn die Ressource existiert, dann in die Parameter width und height tatsächliche Größe zurückangegeben werden. Dadurch ist es ein weiterer Versuch, Daten von der Ressource zu erhalten.

### Sehen Sie auch

[Ressourcen](#), [ObjectCreate\(\)](#), [ObjectSetString\(\)](#), [OBJPROP\\_BITMAP](#)

## ResourceSave

Speichert die Ressource in eine angegebene Datei.

```
bool ResourceSave(  
    const string resource_name    // Ressourcename  
    const string file_name       // Dateiname  
) ;
```

### Parameter

*resource\_name*

[in] Ressourcename muss mit "::" anfangen.

*file\_name*

[in] Dateiname relativ zu MQL5\Files.

### Rückgabewert

Gibt true beim Erfolgsfall zurück, anderenfalls false. Für Erhaltung des [Fehlerkodes](#) muss man die Funktion [GetLastError\(\)](#) aufrufen.

### Bemerkung

Die Funktion überschreibt immer die Datei und erstellt alle temporäre Verzeichnisse in den Dateinamen, wenn es nötig ist.

### Sehen Sie auch

[Ressourcen](#), [ObjectCreate\(\)](#), [PlaySound\(\)](#), [ObjectSetString\(\)](#), [OBJPROP\\_BMPFILE](#)

## SetUserError

Stellt die Vorbestimmte Variable [\\_LastError](#) in Wert [ERR\\_USER\\_ERROR\\_FIRST](#) + user\_error

```
void SetUserError(  
    ushort user_error,    // Fehlernummer  
);
```

### Parameter

*user\_error*

[in] Nummer des [Fehlers](#), festgelegt vom Benutzer.

### Rückgabewert

Keinen Rückgabewert.

### Bemerkung

Nach Einstellung des Fehlers durch die Funktion SetUserError(user\_error), gibt die Funktion [GetLastError\(\)](#) den Wert, gleich [ERR\\_USER\\_ERROR\\_FIRST](#) + user\_error.

### Beispiel:

```
void OnStart()  
{  
    //--- legen wir Fehlernummer 65537=(ERR_USER_ERROR_FIRST +1) fest  
    SetUserError(1);  
    //--- erhalten wir Code des letzten Fehlers  
    Print("GetLastError = ",GetLastError());  
    /*  
    Ergebnis  
    GetLastError = 65537  
    */  
};
```

## SendFTP

Sendet die Datei an die Adresse angegeben im Fenster der Einstellungen im Registerblatt "Publikation".

```
bool SendFTP(  
    string filename,           // Datei für Senden durch ftp  
    string ftp_path=NULL      // Pfad für Ausladen auf ftp-Server  
);
```

### Parameter

*filename*

[in] Name der gesendeten Datei.

*ftp\_path=NULL*

[in] Katalog FTP. Wenn das Verzeichnis nicht angegeben wird, wird das Verzeichnis verwendet, das in Einstellungen beschrieben wird.

### Rückgabewert

Beim Misserfolg gibt false zurück.

### Bemerkung

Die gesendete Datei muss im Ordner *Terminal\_Verzeichnis\MQL5\files* oder in Subordnern sein. Senden wird nicht durchgeführt, wenn es in Einstellungen keine FTP Adresse und/oder kein Zugangspasswort angegeben wird.

Beim Anwenden im [Strategie-Tester](#) wird die SendFTP()-Funktion nicht ausgeführt.



## SendMail

Sendet die Email an die Adresse, gegeben im Fenster der Einstellungen im Registerblatt "Post".

```
bool SendMail(  
    string subject,      // Kopf  
    string some_text     // Text der Email  
);
```

### Parameter

*subject*

[in] Kopf der Email.

*some\_text*

[in] Koerper der Email.

### Rückgabewert

true - wenn der Brief In Sendequeue gestellt wird, sonst false.

### Bemerkung

Senden kann in Einstellungen verboten sein, auch kann Emailadresse nicht angegeben werden. Für Fehlerinformation muss man die Funktion [GetLastError\(\)](#) aufrufen.

Beim Anwenden im [Strategie-Tester](#) wird die SendMail()-Funktion nicht ausgeführt.

## SendNotification

Sendet eine Benachrichtigung zu mobilen Terminals, deren MetaQuotes-IDs in das Optionen-Fenster auf der Registerkarte "Benachrichtigungen" angegeben sind.

```
bool SendNotification(  
    string text          // Text der Benachrichtigung  
);
```

### Parameter

*text*

[in] Text der Benachrichtigung. Die Benachrichtigung sollte nicht länger als 255 Zeichen sein.

### Rückgabewert

true, wenn Benachrichtigung erfolgreich aus dem Terminal gesendet ist, ansonsten gibt false zurück. Bei der Überprüfung nach dem Fehlschlagen der benachrichtigen [GetLastError\(\)](#) kann ein der folgenden Fehler ausgegeben:

- 4515 - ERR\_NOTIFICATION\_SEND\_FAILED,
- 4516 - ERR\_NOTIFICATION\_WRONG\_PARAMETER,
- 4517 - ERR\_NOTIFICATION\_WRONG\_SETTINGS,
- 4518 - ERR\_NOTIFICATION\_TOO\_FREQUENT.

### Bemerkung

Für die Funktion SendNotification() sind enge Einschränkungen für die Verwendung gesetzt: nicht mehr als 2 Anrufe pro Sekunde und nicht mehr als 10 Anrufe pro Minute. Die Überwachung der Häufigkeit der Nutzung ist dynamisch, und die Funktion kann bei der Verletzung deaktiviert werden.

Beim Anwenden im [Strategie-Tester](#) wird die SendNotification()-Funktion nicht ausgeführt.

## Sleep

Funktion verzögert die Ausführung des laufenden Experten oder des Scripts für bestimmtes Intervall.

```
void Sleep(  
    int milliseconds    // Intervall  
);
```

### Parameter

*milliseconds*

[in] Verzögerungsintervall in Milisekunden.

### Rückgabewert

Keinen Rückgabewert.

### Bemerkung

Die Funktion Sleep() kann aus Benutzerindikatoren nicht aufgerufen werden, denn Intervalle werden im Interface Thread ausgeführt und müssen es nicht hemmen. Die Funktion hat die eingebaute Prüfung der Halteflagge des Experten jede 0.1 Sekunde.

## TerminalClose

Sendet dem Terminal den Befehl, Operation zu beenden.

```
bool TerminalClose(
    int ret_code    // Abschlusscode des Kundenterminals
);
```

### Parameter

*ret\_code*

[in] Rückgabewert, der vom Prozess des Kundenterminals beim Operationsabschluss zurückgegeben wird.

### Rückgabewert

Gibt true beim Erfolg zurück, sonst false.

### Bemerkung

Funktion TerminalClose() stoppt die Terminaloperation nicht sofort, sie sendet dem Terminal den Befehl, Arbeit abzuschliessen.

Im Ratgebercode, der TerminalClose() aufgerufen hat, müssen alle Vorbereitungen gemacht werden, um Operation sofort zu beenden (ZB alle früher geöffnete Dateien müssen routinemäßig geschlossen werden). Sofort nach Aufruf dieser Funktion muss die [Anweisung return](#) .

Parameter *ret\_code* ermöglicht den notwendigen Rückgabekode anzugeben, um Gründe des Programmabschlusses der Terminaloperation zu analysieren, wenn das Terminal aus der Befehlszeile gestartet wird.

### Beispiel:

```
//--- input parameters
input int  tiks_before=500; // Zahl der Ticks bis Abschluss
input int  pips_to_go=15;   // Distanz in Pips
input int  seconds_st=50;   // Zahl der Sekunden gegeben dem Experten
//--- globals
datetime   launch_time;
int        tick_counter=0;
//+-----+
//| Expert deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
    //---
    Print(__FUNCTION__, " reason code = ", reason);
    Comment("");
};
//+-----+
//| Expert tick function |
//+-----+
void OnTick()
```

```

{
    static double first_bid=0.0;
    MqlTick      tick;
    double       distance;
//---
    SymbolInfoTick(_Symbol,tick);
    tick_counter++;
    if(first_bid==0.0)
    {
        launch_time=tick.time;
        first_bid=tick.bid;
        Print("first_bid = ",first_bid);
        return;
    }
//--- Preisgang in Punkten
    distance=(tick.bid-first_bid)/_Point;
//--- Geben wir die Meldung aus, um Arbeit des Experten zu befolgen
    string comm="Vom Start:\r\n\x25CF vergang es Sekunden: "+
        IntegerToString(tick.time-launch_time)+" ;"+
        "\r\n\x25CF erhaltene Ticks: "+(string)tick_counter+" ;"+
        "\r\n\x25CF Preis ging in Punkten: "+StringFormat("%G",distance);
    Comment(comm);
//--- Abschnitt für Pruefung der Bedingungen für Terminalabschluss
    if(tick_counter>=tiks_before)
        TerminalClose(0);    // Ende nach Tickcounter
    if(distance>pips_to_go)
        TerminalClose(1);    // gehen oben um Pipszahl pips_to_go
    if(distance<-pips_to_go)
        TerminalClose(-1);   // gehen unten um Pipszahl pips_to_go
    if(tick.time-launch_time>seconds_st)
        TerminalClose(100);  // Beenden laut Timeout
//---
};

```

Sehen Sie auch

[Programmausführung](#), [Ausführungsfehler](#), [Deinitialisierungsguende](#)

## TesterStatistics

Die Funktion gibt den Wert der statistischen Parameter, der nach Testergebnisse berechnet ist, zurück.

```
double TesterStatistics(  
    ENUM_STATISTICS statistic_id    // ID  
);
```

### Parameter

*statistic\_id*

[in] Identifikator des statistischen Parameter aus der [ENUM\\_STATISTICS](#) Aufzählung.

### Rückgabewert

Der Wert des statistischen Parameter des Testergebnisse.

### Bemerkung

Die Funktion kann innerhalb [OnTester\(\)](#) oder [OnDeinit\(\)](#) im Tester genannt werden. In anderen Fällen ist das Ergebnis undefiniert.

## TesterWithdrawal

Sonderfunktion für Emulation der Geldabhebungen beim Testen. Kann in einigen Systemen der Kapitalverwaltung verwendet werden.

```
bool TesterWithdrawal(  
    double money    // Anzahl der abgehebener Summe  
);
```

### Parameter

*money*

[in] Anzahl der zu vom Konto abzugehebener Summe (in Depositwährung)

### Rückgabewert

Gibt true beim Erfolgsfall zurück, anderenfalls false.

## WebRequest

Die Funktion sendet eine HTTP-Anfrage an den Server. Es gibt drei Varianten der Funktion:

1. **Senden einfache Anfrage** wie "Schlüssel=Wert" mit dem Header Content-Type: application/x-www-form-urlencoded.

```
int WebRequest(
    const string    method,           // HTTP-Methode
    const string    url,              // URL-Adresse
    const string    cookie,           // Cookie
    const string    referer,          // Referer
    int             timeout,           // Timeout
    const char      &data[],           // Array des Körpers von HTTP-Nachrichten
    int             data_size,         // Größe des data[]-Arrays in Bytes
    char            &result[],         // Array mit Daten aus Serverantwort
    string          &result_headers  // Headers von Serverantwort
);
```

2. **Senden jede Anfrage** mit einem eigenen Set von Headers für weitere flexible Interaktion mit verschiedenen Web-Services.

```
int WebRequest(
    const string    method,           // HTTP-Methode
    const string    url,              // URL-Adresse
    const string    headers,          // Headers
    int             timeout,           // Timeout
    const char      &data[],           // Array des Körpers von HTTP-Nachrichten
    char            &result[],         // Array mit Daten aus Serverantwort
    string          &result_headers  // Headers von Serverantwort
);
```

### Parameter

*method*

[in] HTTP-Methode.

*url*

[in] URL-Adresse

*headers*

[in] Headers vom Type "Schlüssel=Wert", die durch Zeilenumbruch "\r\n" getrennt sind.

*cookie*

[in] Wert von Cookie.

*referer*

[in] Wert von Referer der HTTP-Anfrage.

*timeout*

[in] Timeout in Millisekunden.



*data[]*

[in] Array der Körperdaten von HTTP-Nachrichten.

*data\_size*

[in] Größe des data[]-Arrays.

*result[]*

[out] Array mit Daten von Serverantwort

*result\_headers*

[out] Headers von Serverantwort.

### Rückgabewert

Antwortcode des HTTP-Servers oder -1 bei einem Fehler.

### Bemerkung

Um die WebRequest-Funktion zu verwenden, schreiben Sie die Serveradressen in der Liste der erlaubten URL auf der Registerkarte "Experten" im Fenster "Einstellungen". Serverport wird automatisch ausgewählt auf der Grundlage des Protokolls - 80 für "http://" und 443 für "https://".

Die WebRequest()-Funktion ist synchron, das heißt, sie unterbricht die Ausführung des Programms und wartet auf eine Antwort von dem angefragten Server. Da die Verzögerung beim Empfang einer Antwort auf die gesendete Anfrage groß sein kann, ist die Funktion für Anrufe aus den Indikatoren verboten, als Indikatoren arbeiten in einem einzigen Datenfluß, der für alle Indikatoren und Charts auf dieses Symbol gesamt ist. Verzögerung in der Ausführung des Indikators auf einem des Symbolcharten kann Aktualisierung aller Charts für dieses Symbol stoppen.

Die Funktion kann nur von Experten und Skripts aufgerufen werden, da sie in ihrer eigenen Ausführungsumfang arbeiten. Wenn die Funktion aus einem Indikator aufgerufen wird, gibt [GetLastError\(\)](#) den Fehler 4014 - "Die Funktion ist nicht erlaubt" zurück.

Beim Anwenden im [Strategie-Tester](#) wird die WebRequest()-Funktion nicht ausgeführt.

### Beispiel der 1. Version der Funktion WebRequest():

```
void OnStart()
{
    string cookie=NULL,headers;
    char post[],result[];
    int res;
    //--- für die Arbeit mit dem Server, fügen wir URL "https://www.google.com/finance"
    //--- in der Liste der erlaubten URLs hinzu (Hauptmenü->Service->Einstellungen, Regist
    string google_url="https://www.google.com/finance";
    //--- den letzten Fehlercode rücksetzen
    ResetLastError();
    //--- HTML-Seite aus Google Finance herunterladen
    int timeout=5000; //--- Timeout weniger als 1000 (1 Sek.) ist nicht ausreichend bei e
    res=WebRequest("GET",google_url,cookie,NULL,timeout,post,0,result,headers);
    //--- Fehlerprüfung
    if(res===-1)
    {
```

```

Print("Fehler in WebRequest. Fehlercode =",GetLastError());
//--- Vielleicht ist diese URL in der Liste nicht geschrieben; eine Meldung über
MessageBox("Die Adresse '"+google_url+"' muss in der Liste der erlaubten URLs au
}
else
{
//--- erfolgreiches Herunterladen
PrintFormat("Datei erfolgreich heruntergeladen, Dateigröße =%d Bytes.",ArraySize
//--- Daten in einer Datei speichern
int filehandle=FileOpen("GoogleFinance.htm",FILE_WRITE|FILE_BIN);
//--- Fehlerprüfung
if(filehandle!=INVALID_HANDLE)
{
//--- die Inhalte des Arrays result[] in der Datei speichern
FileWriteArray(filehandle,result,0,ArraySize(result));
//--- die Datei schließen
FileClose(filehandle);
}
else Print("Ein Fehler in FileOpen. Fehlercode =",GetLastError());
}
}

```

### Beispiel der zweiten Version der Funktion WebRequest():

```

#property link      "http://www.mql5.com"
#property version   "1.00"
#property strict
#property script_show_inputs
#property description "Ein Beispiel für ein Skript, das eine Nachricht des Benutzers '
#property description "auf die Wand auf mql5.com"

input string InpLogin   ="";           //Ihr Konto auf MQL5.com
input string InpPassword="";          //Passwort für Ihr Konto
input string InpFileName="EURUSDM5.png"; //Ein Bild im Ordner MQL5/Files/
input string InpFileType="image/png";   //Richtiges MIME-Typ des Bildes
//+-----+
//| Publikation einer Nachricht mit einem Bild auf mql5.com Wand |
//+-----+

bool PostToNewsFeed(string login,string password,string text,string filename,string fi
{
    int   res;    // Das Ergebnis der Operation kommt hier
    char  data[]; // Array mit Daten der Sendung von POST-Anfragen
    char  file[]; // Lesen das Bild hier
    string str="Login="+login+"&Password="+password;
    string auth,sep="-----Jyecslin9mp8RdKV"; // Trennzeichen von Daten multipart
//--- Es gibt eine Datei - versuchen wir es zu lesen
    if(filename!=NULL && filename!="")
    {

```

```

res=FileOpen(filename,FILE_READ|FILE_BIN);
if(res<0)
{
    Print("Fehler bei der Öffnung der Datei \""+filename+"\"");
    return(false);
}
//--- Lesen die Dateidaten
if(FileReadArray(res,file)!=FileSize(res))
{
    FileClose(res);
    Print("Fehler beim Lesen der Datei \""+filename+"\"");
    return(false);
}
//---
FileClose(res);
}
//--- Bilden wir den Körper einer POST-Anfrage für die Autorisierung
ArrayResize(data,StringToCharArray(str,data,0,WHOLE_ARRAY,CP_UTF8)-1);
//--- den Fehlercode rücksetzen
ResetLastError();
//--- Anforderung für die Autorisierung
res=WebRequest("POST","https://www.mql5.com/ru/auth_login",NULL,0,data,data,str);
//--- Wenn die Autorisierung fehlschlägt ist
if(res!=200)
{
    Print("Fehler bei der Autorisierung #"+(string)res+", LastError="+ (string)GetLastError());
    return(false);
}
//--- Autorisierung-Cookie im Header vom Serverantwort
res=StringFind(str,"Set-Cookie: auth=");
//--- Wenn das Cookie nicht gefunden wird, Fehler melden
if(res<0)
{
    Print("Fehler, Autorisierungsdaten in der Server-Antwort nicht gefunden (Überprüfen Sie die URL)");
    return(false);
}
//--- Autorisierungsdaten speichern und einen Header für nachfolgende Anfragen bilden
auth=StringSubstr(str,res+12);
auth="Cookie: "+StringSubstr(auth,0,StringFind(auth,";")+1)+"\r\n";
//--- Wenn es eine Datendatei gibt, senden wir sie an den Server
if(ArraySize(file)!=0)
{
    //--- Wir bilden das Hauptteil der Anfrage
    str="--"+sep+"\r\n";
    str+="Content-Disposition: form-data; name=\"attachedFile_imagesLoader\"; filename="+file+"\r\n";
    str+="Content-Type: "+filetype+"\r\n\r\n";
    res =StringToCharArray(str,data);
    res+=ArrayCopy(data,file,res-1,0);
    res+=StringToCharArray("\r\n--"+sep+"--\r\n",data,res-1);
}

```

```

ArrayResize(data,ArraySize(data)-1);
//--- Wir bilden das Header der Anfrage
str=auth+"Content-Type: multipart/form-data; boundary="+sep+"\r\n";
//--- den Fehlercode rücksetzen
ResetLastError();
//--- Anfrage eine Bilddatei an den Server zu senden
res=WebRequest("POST","https://www.mql5.com/upload_file",str,0,data,data,str);
//--- Ergebnisse der Anfrage überprüfen
if(res!=200)
{
    Print("Fehler beim Senden einer Datei auf den Server #"+(string)res+", LastE
    return(false);
}
//--- Bekommen wir einen Link zum Bild, das wir auf den Server hochgeladen haben
str=CharArrayToString(data);
if(StringFind(str,{"\Url\":"})==0)
{
    res    =StringFind(str,"\"",8);
    filename=StringSubstr(str,8,res-8);
    //--- Beim Ladungsfehler wird ein leerer Link zurückgegeben
    if(filename=="")
    {
        Print("Senden einer Datei auf den Server fehlgeschlagen");
        return(false);
    }
}
}
//--- Wir bilden das Hauptteil der Anfrage um eine Nachricht auf den Server zu veröffentli
str="--"+sep+"\r\n";
str+="Content-Disposition: form-data; name=\"content\""\r\n\r\n";
str+=text+"\r\n";
//--- Sprachen der Nachricht auf mql5.com
str="--"+sep+"\r\n";
str+="Content-Disposition: form-data; name=\"AllLanguages\""\r\n\r\n";
str+="on\r\n";
//--- Wenn das Bild auf den Server geladen wurde - geben wir seinen Link weiter
if(ArraySize(file)!=0)
{
    str="--"+sep+"\r\n";
    str+="Content-Disposition: form-data; name=\"attachedImage_0\""\r\n\r\n";
    str+=filename+"\r\n";
}
//--- Die letzte Zeile der mehrteiligen-Anfrage
str="--"+sep+"--\r\n";
//--- Körper der POST-Anfrage in einer Linie bilden
StringToCharArray(str,data,0,WHOLE_ARRAY,CP_UTF8);
ArrayResize(data,ArraySize(data)-1);
//--- Header der Anfrage bereiten
str=auth+"Content-Type: multipart/form-data; boundary="+sep+"\r\n";

```

```

//--- Senden Anfrage auf Veröffentlichung der Nachricht auf dem Wand auf mql5.com
    res=WebRequest("POST","https://www.mql5.com/en/users/"+login+"/wall",str,0,data,data)
//--- Bei erfolgreicher Publikationen gibt true zurück
    return(res==200);
}
//+-----+
//| Script program start function                                     |
//+-----+
void OnStart()
{
//--- Veröffentlichen wir eine Nachricht am mql5.com mit einem Bild, dessen Weg wir au
    PostToNewsFeed(InpLogin,InpPassword,"Überprüfen der erweiterten Version der Funktio
        "(Diese Nachricht wurde durch dem Skript WebRequest.mq5 hinzugefügt)
}
//+-----+

```

## ZeroMemory

Funktion nullt die Variable aus, die ihr durch Referenz übertragen wird.

```
void ZeroMemory(  
    void & variable    // ausgeNULLte Variable  
);
```

### Parameter

*variable*

[in] [out] Variable, übertragen durch Referenz, die ausgeNULLt werden muss (initialisiert durch Null-Werte).

### Rückgabewert

Keinen Rückgabewert.

### Bemerkung

Wenn der Funktionsparameter Zeile ist, wird der Aufruf für sie dem Wert NULL gleich.  
für einfache Typen und ihre Felder, aber auch für Strukturen/Klassen, bestehenden aus diesen Typen ist es bloß AusNULLen.  
für Objekte mit Zeilen und dynamischen Feldern, wird ZeroMemory() für jedes Glied aufgerufen.  
für Felder, die nicht vom Modifikator const geschützt sind, werden alle Elementen ausgeNULLt.  
für Felder komplizierter Objekten wird ZeroMemory() für jedes Element aufgerufen.

Funktion ZeroMemory() kann nicht für Klassen mit geschützten [Gliedern](#) oder [Vererbung](#) angewendet werden.

## Gruppe der Funktionen für Arbeit mit Arrays

Arrays können nicht mehr als vierdimensional sein. Jede Dimension wird von Null bis *Dimensions\_Größe-1* indiziert. Im Einzelfall des eindimensionalen Arrays aus 50 Elementen wird die Anwendung zum ersten Element als `array[0]`, zum letzten - `array[49]` aussehen.

| Funktion                                | Handlung                                                                                                                                                                     |
|-----------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#"><u>ArrayBsearch</u></a>     | Gibt Index des ersten gefundenen Elementes in der ersten Dimension des Arrays                                                                                                |
| <a href="#"><u>ArrayCopy</u></a>        | Kopiert ein Array in das andere Array                                                                                                                                        |
| <a href="#"><u>ArrayCompare</u></a>     | Gibt das Ergebnis eines Vergleichs um zwei Arrays von <a href="#"><u>einfachen Typen</u></a> oder angepassten Strukturen ohne <a href="#"><u>komplexe Objekte</u></a> zurück |
| <a href="#"><u>ArrayFree</u></a>        | Befreit Puffer jedes dynamischen Arrays und stellt die Größe der ersten Null-Dimension auf 0.                                                                                |
| <a href="#"><u>ArrayGetAsSeries</u></a> | Prüft die Richtung des Arrayindizierens                                                                                                                                      |
| <a href="#"><u>ArrayInitialize</u></a>  | Stellt alle Elementen des numerischen Arrays in eine Größe ein                                                                                                               |
| <a href="#"><u>ArrayFill</u></a>        | Füllt das numerische Array mit den angegeben Wert                                                                                                                            |
| <a href="#"><u>ArrayIsSeries</u></a>    | Prüft - ob das Array Timeserie ist                                                                                                                                           |
| <a href="#"><u>ArrayIsDynamic</u></a>   | Prüft - ob das Array dynamisches Array ist                                                                                                                                   |
| <a href="#"><u>ArrayMaximum</u></a>     | Suche des Elementes mit maximalem Wert                                                                                                                                       |
| <a href="#"><u>ArrayMinimum</u></a>     | Suche des Elementes mit minimalem Wert                                                                                                                                       |
| <a href="#"><u>ArrayRange</u></a>       | Gibt die Zahl der Elementen in der gegebenen Dimension des Arrays zurück                                                                                                     |
| <a href="#"><u>ArrayResize</u></a>      | Stellt die neue Größe in der ersten Dimension des Arrays ein                                                                                                                 |
| <a href="#"><u>ArraySetAsSeries</u></a> | Stellt die Richtung der Indizierung im Array ein                                                                                                                             |
| <a href="#"><u>ArraySize</u></a>        | Gibt die Zahl von Elementen im Array zurück                                                                                                                                  |
| <a href="#"><u>ArraySort</u></a>        | Sortieren der numerischen Arrays nach der ersten Dimension                                                                                                                   |

## ArrayBsearch

Sucht im eindimensionalen numerischen Array, der ansteigend sortiert wurde, den angegebenen Wert.

**für Suchen im Array des Typs double**

```
int ArrayBsearch(  
    const double&    array[],    // das Array für die Suche  
    double          value        // was wird gesucht  
);
```

**für Suche im Array des Typs float**

```
int ArrayBsearch(  
    const float&     array[],    // das Array für die Suche  
    float            value        // was wird gesucht  
);
```

**für Suche im Array des Typs long**

```
int ArrayBsearch(  
    const long&      array[],    // das Array für die Suche  
    long             value        // was wird gesucht  
);
```

**für Suche im Array des Typs int**

```
int ArrayBsearch(  
    const int&       array[],    // Array für die Suche  
    int              value        // was wird gesucht  
);
```

**für Suche im Array des Typs short**

```
int ArrayBsearch(  
    const short&     array[],    // Array für die Suche  
    short            value        // was wird gesucht  
);
```

**für Suche im Array des Typs char**

```
int ArrayBsearch(  
    const char&      array[],    // Array für die Suche  
    char             value        // was wird gesucht  
);
```

### Parameter

*array[]*

[in] Numerisches Array für die Suche.

*value*

[in] Wert für die Suche.



**Rückgabewert**

Gibt Index des gefundenen Elementes zurück. Wenn der gesuchte Wert nicht gefunden wird, wird Index des Elementes, der den nächsten Wert hat.

**Bemerkung**

Binäre Suche verarbeitet sortierte Arrays nur. Für Sortieren des numerischen Arrays wird die Funktion [ArraySort\(\)](#) verwendet.

**Beispiel:**

```

#property description "Skript auf der RSI Indikator wird im Chart Fenster angezeigt"
#property description "die Daten wie oft der Markt in eine überkauft"
#property description "und überverkauft Zonen in den angegebenen Zeitraum war."
//--- Beim Starten des Skripts zeigen Sie den Fenster mit den Eingabeparametern an
#property script_show_inputs
//--- Eingabeparameters
input int          InpMAPeriod=14;                // Periode des gleitenden
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE; // Preistyp
input double        InpOversoldValue=30.0;        // Überverkauft Ebene
input double        InpOverboughtValue=70.0;      // Überkauft Ebene
input datetime      InpDateStart=D'2012.01.01 00:00'; // Das Anfangsdatum der Ar
input datetime      InpDateFinish=D'2013.01.01 00:00'; // das Abschlußdatum der Z
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    double rsi_buff[]; // das Array der Indikatorwerte
    int size=0;        // die Größe des Arrays
//--- erhalten Sie die Handle des RSI Indikators
    ResetLastError();
    int rsi_handle=iRSI(Symbol(),Period(),InpMAPeriod,InpAppliedPrice);
    if(rsi_handle==INVALID_HANDLE)
    {
        //--- Es misslang, Handle des Indikators zu bekommen
        PrintFormat("Fehler beim Erhalten die Handle des Indikators. Fehlercode = %d",GetLa
        return;
    }
//--- Wir befinden uns im Zyklus, bis der Indikator alle Werten berechnen wird
    while(BarsCalculated(rsi_handle)==-1)
    {
        //--- Wir gehen hinaus, wenn der Benutzer die Arbeit des Skripts zwangsläufig be
        if(IsStopped())
            return;
        //--- die Verzögerungszeit, damit der Indikator geschafft, seine Werte zu berech
        Sleep(10);
    }
//--- kopieren Sie die Indikatorwerte für eine angegebene Period
    ResetLastError();
    if(CopyBuffer(rsi_handle,0,InpDateStart,InpDateFinish,rsi_buff)==-1)
    {
        PrintFormat("Es misslang, die Indikatorwerte zu kopieren. Fehlercode = %d",GetLa
        return;
    }
//--- erhalten Sie die Größe des Arrays
    size=ArraySize(rsi_buff);
//--- sortieren Sie das Array
    ArraySort(rsi_buff);
//--- erkennen Sie welches Prozent der Zeit sich den Markt in der überverkauft Zone be
    double ovs=(double)ArrayBsearch(rsi_buff,InpOversoldValue)*100/(double)size;
//--- erkennen Sie welches Prozent der Zeit sich den Markt in der überkauft Zone befand
    double ovb=(double)(size-ArrayBsearch(rsi_buff,InpOverboughtValue))*100/(double)si
//--- bilden Sie die Zeilen für die Schlussfolgerung der Daten
    string str="Seit "+TimeToString(InpDateStart,TIME_DATE)+" bis "
        +TimeToString(InpDateFinish,TIME_DATE)+" der Markt ist in:";
    string str_ovb="den überkauft Zone "+DoubleToString(ovb,2)+"% Zeit";
    string str_ovs="den überverkauft Zone "+DoubleToString(ovs,2)+"% Zeit";
//--- zeigen Sie die Daten in der Chart
    CreateLabel("top",5,60,str,clrDodgerBlue);
    CreateLabel("overbought",5,35,str_ovb,clrDodgerBlue);
    CreateLabel("oversold",5,10,str_ovs,clrDodgerBlue);

```

```

//--- zeichnen Sie die Chart
    ChartRedraw(0);
//--- die Verzögerung
    Sleep(10000);
}
//+-----+
//| Zeigt den Kommentar in der unteren linken Ecke der Chart |
//+-----+
void CreateLabel(const string name,const int x,const int y,
                const string str,const color clr)
{
//--- Label erstellen
    ObjectCreate(0,name,OBJ_LABEL,0,0,0);
//--- Bindung von Label zu dem linken unteren Ecke
    ObjectSetInteger(0,name,OBJPROP_CORNER,CORNER_LEFT_LOWER);
//--- ändern Sie die Position der Ankerpunkt
    ObjectSetInteger(0,name,OBJPROP_ANCHOR,ANCHOR_LEFT_LOWER);
//--- die Länge der X-Achse von der Ankerpunkt
    ObjectSetInteger(0,name,OBJPROP_XDISTANCE,x);
//--- die Länge der Y-Achse von der Ankerpunkt
    ObjectSetInteger(0,name,OBJPROP_YDISTANCE,y);
//--- der Text-Label
    ObjectSetString(0,name,OBJPROP_TEXT,str);
//--- die Textfarbe
    ObjectSetInteger(0,name,OBJPROP_COLOR,clr);
//--- die Textgröße
    ObjectSetInteger(0,name,OBJPROP_FONTSIZE,12);
}

```

## ArrayCopy

Kopiert ein Array in das andere Array.

```
int ArrayCopy(
    void&          dst_array[],      // Zielarray für Kopieren
    const void&    src_array[],      // Das Array von dem kopiert wird
    int            dst_start=0,      // Index, mit dem in Rezipienten geschrieben wird
    int            src_start=0,      // Index, mit dem Kopieren von der Quelle beginnt
    int            count=WHOLE_ARRAY // Anzahl der Elemente
);
```

### Parameter

*dst\_array[]*

[out] Array-Rezipient.

*src\_array[]*

[in] Ausgangs-Array.

*dst\_start=0*

[in] Anfangsindex für Array-Rezipienten. Standardwert des Anfangsindexes ist - 0.

*src\_start=0*

[in] Anfangsindex für Ausgangs-Array. Standardwert des Anfangsindexes ist - 0.

*count=WHOLE\_ARRAY*

[in] Anzahl der Elemente für Kopieren. Als Voreinstellung wird das ganze Array kopiert. (count=WHOLE\_ARRAY).

### Rückgabewert

Gibt die Anzahl der kopierten Elemente zurück.

### Bemerkung

Wenn  $\text{count} < 0$  oder  $\text{count} > \text{src\_size} - \text{src\_start}$ , wird das Rest des Arrays kopiert. Arrays werden von links nach rechts kopiert. Für serielle Arrays wird die Anfangsposition unter Berücksichtigung des Kopierens von links nach rechts korrekt Neubestimmt. Wenn das Array in sich selbst kopiert wird, ist das Ergebnis nicht bestimmt.

Wenn die Arrays verschiedener Typen angehören, wird beim Kopieren der Versuch, jedes Element des Ausgangsarrays zum Typ des Array-Rezipienten zu reduzieren. Ein Zeilenarray kann nur in ein Zeilenarray kopiert werden. Arrays der [Klassen und Strukturen](#), die Objekte enthalten, die initialisiert werden müssen, werden nicht kopiert. Array der Strukturen kann nur ins Array desselben Typs kopiert werden.

Für dynamische Arrays mit Indexierung wie in [Timeseries](#) wird das Empfänger-Array automatisch bis die Menge der kopierenden Daten vergrößert (wenn die Menge der kopierenden Daten größer als das Array ist). Empfänger-Array wird nicht automatisch verkleinert.

### Beispiel:

```
#property description "Der Indikator hervorhebt die Kerzen, die lokal sind"
```

```

#property description "Maximum und Minimum. Die Länge des Intervalls für Aufsuchen"
#property description "die Extremums kann man durch die Eingabeparameters einstellt w
//--- Indikator Einstellungen
#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots 1
//---- plot
#property indicator_label1 "Extremums"
#property indicator_type1 DRAW_COLOR_CANDLES
#property indicator_color1 clrLightSteelBlue,clrRed,clrBlue
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//--- vordefinierte Konstante
#define INDICATOR_EMPTY_VALUE 0.0
//--- Eingabeparameters
input int InpNum=4; // Die Länge des Halb-Intervalls
// --- Indikator Puffers
double ExtOpen[];
double ExtHigh[];
double ExtLow[];
double ExtClose[];
double ExtColor[];
//--- globale Variablen
int ExtStart=0; // der Index der ersten Kerze, die kein Extremum ist
int ExtCount=0; // Anzahl der Kerzen, nicht den Extremums in diesen Intervall
//+-----+
//| Zeichnen die Kerzen, nicht Extremums |
//+-----+
void FillCandles(const double &open[],const double &high[],
                 const double &low[],const double &close[])
{
//--- Zeichnen Sie die Kerzen
ArrayCopy(ExtOpen,open,ExtStart,ExtStart,ExtCount);
ArrayCopy(ExtHigh,high,ExtStart,ExtStart,ExtCount);
ArrayCopy(ExtLow,low,ExtStart,ExtStart,ExtCount);
ArrayCopy(ExtClose,close,ExtStart,ExtStart,ExtCount);
; }
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- Bindung von Array zum Indikator Puffer
SetIndexBuffer(0,ExtOpen);
SetIndexBuffer(1,ExtHigh);
SetIndexBuffer(2,ExtLow);
SetIndexBuffer(3,ExtClose);
SetIndexBuffer(4,ExtColor,INDICATOR_COLOR_INDEX);
//--- erstellen Sie die Wert, die nicht sichtbar sein werden

```

```

    PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,INDICATOR_EMPTY_VALUE);
//--- erstellen Sie die Name von Indikator-Puffers für die Anzeige im Daten-Fenster
    PlotIndexSetString(0,PLOT_LABEL,"Open;High;Low;Close");
//---
    return(INIT_SUCCEEDED);
; }
//+-----+
//| Custom indicator iteration function |
//+-----+

int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{
//--- stellen Sie die gerade Richtung der Indexierung in Zeitreihen ein
    ArraySetAsSeries(open,false);
    ArraySetAsSeries(high,false);
    ArraySetAsSeries(low,false);
    ArraySetAsSeries(close,false);
//--- Variable zur Berechnung den Anfang des Bars
    int start=prev_calculated;
//--- für die ersten InpNum*2 Bars wird die Berechnung nicht durchgeführt
    if(start==0)
    {
        start+=InpNum*2;
        ExtStart=0;
        ExtCount=0;
    ; }
//--- Wenn sich nur eine Bar gebildet hat, so werden wir den nächsten potentiellen Ext
    if(rates_total-start==1)
        start--;
//--- der Index der Bar, den wir auf Extremum prüfen werden
    int ext;
//--- die Zyklus von die Berechnung der Indikatorwerte
    for(int i=start;i<rates_total-1;i++)
    {
        //--- vornherein an der i-ten Bar ohne Zeichnung
        ExtOpen[i]=0;
        ExtHigh[i]=0;
        ExtLow[i]=0;
        ExtClose[i]=0;
        //--- der Extremum Index für die Überprüfung
        ext=i-InpNum;
    }
}

```

```

//--- die Überprüfung für das lokale Maximum
if(IsMax(high,ext))
{
    //--- bezeichnen Sie die Extremum Kerze
    ExtOpen[ext]=open[ext];
    ExtHigh[ext]=high[ext];
    ExtLow[ext]=low[ext];
    ExtClose[ext]=close[ext];
    ExtColor[ext]=1;
    //--- die übrigen Kerzen werden wir bis zu Extremum mit der neutralen Farbe k
    FillCandles(open,high,low,close);
    //--- ändern Sie die Variablenwerte
    ExtStart=ext+1;
    ExtCount=0;
    //--- gehen Sie zum nächsten Iteration
    continue;
};
//--- Überprüfung für das lokale Minimum
if(IsMin(low,ext))
{
    //--- bezeichnen Sie die Extremum Kerze
    ExtOpen[ext]=open[ext];
    ExtHigh[ext]=high[ext];
    ExtLow[ext]=low[ext];
    ExtClose[ext]=close[ext];
    ExtColor[ext]=2;
    //--- die übrigen Kerzen werden wir bis zu Extremum mit der neutralen Farbe k
    FillCandles(open,high,low,close);
    //--- ändern Sie die Variablenwerte
    ExtStart=ext+1;
    ExtCount=0;
    //--- gehen Sie zum nächsten Iteration
    continue;
};
// vergrößern Sie die Anzahl nicht den Extremums in diesem Intervall
ExtCount++;
};
//--- den Wert prev_calculated für den nächsten Anruf zurückgeben
return(rates_total);
};
//+-----+
//| Ob das aktuelle Element des Arrays ein lokales Maximum ist |
//+-----+
bool IsMax(const double &price[],const int ind)
{
    //--- Variable von dem Intervall-Anfang
    int i=ind-InpNum;
    //--- Variable von dem Intervall-Abschluß
    int finish=ind+InpNum+1;

```

```

//--- die Überprüfung für die erste Hälfte des Intervalls
for(;i<ind;i++)
{
    if(price[ind]<=price[i])
        return(false);
};}

//--- die Überprüfung für die zweite Hälfte des Intervalls
for(i=ind+1;i<finish;i++)
{
    if(price[ind]<=price[i])
        return(false);
};}

//--- es ist Extremum
return(true);
};}

//+-----+
//| Ob das aktuell Element des Arrays ein lokales Minimum ist |
//+-----+
bool IsMin(const double &price[],const int ind)
{
    //--- Variable von dem Intervall-Anfang
    int i=ind-InpNum;
    //--- Variable von dem Intervall-Abschluß
    int finish=ind+InpNum+1;
    //--- die Überprüfung für die erste Hälfte des Intervalls
    for(;i<ind;i++)
    {
        if(price[ind]>=price[i])
            return(false);
    };}

    //--- die Überprüfung für die zweite Hälfte des Intervalls
    for(i=ind+1;i<finish;i++)
    {
        if(price[ind]>=price[i])
            return(false);
    };}

    //--- es ist Extremum
    return(true);
};}

```



## ArrayCompare

Gibt das Ergebnis eines Vergleichs von zwei Arrays des gleichen Typs zurück. Es kann verwendet werden, um Arrays von [einfachen Typen](#) oder angepassten Strukturen ohne [komplexe Objekte](#) zu vergleichen - d.h. die keine [Zeichen](#), [dynamische Arrays](#) von Klassen oder anderen Strukturen mit komplexe Objekten enthalten.

```
int ArrayCompare(  
    const void& array1[],           // das erste Array  
    const void& array2[],           // das zweite Array  
    int start1=0,                   // das Anfangsvorspannung im ersten Array  
    int start2=0,                   // das Anfangsvorspannung im zweiten Array  
    int count=WHOLE_ARRAY           // Die Anzahl der Elemente für den Vergleich  
);
```

### Parameters

*array1[]*

[in] Das erste Array.

*array2[]*

[in] Das zweite Array.

*start1=0*

[In] Der Ausgangsindex des Elementes im ersten Array, mit dem der Vergleich beginnen wird. Als Voreinstellung der Startindex - 0.

*start2=0*

[In] Der Ausgangsindex des Elementes im zweiten Array, mit dem der Vergleich beginnen wird. Als Voreinstellung der Startindex - 0.

*count=WHOLE\_ARRAY*

[in] Die Anzahl der Elemente, die man vergleichen muss. Als Voreinstellung beteiligt alle Elemente beider Arrays in dem Vergleich (count=[WHOLE\\_ARRAY](#)).

### Der Rückgabewert

- -1, wenn array1[] weniger als array2[]
- 0, wenn array1[] und array2[] sind gleich
- 1, wenn array1[] mehr als array2[]
- 2, Beim Auftreten des Fehlers wegen der Inkompatibilität der Typen der verglichenen Arrays, oder wenn es start1, start2 oder count Werte gibt, die zum Ausgang aus dem Grenzen von Array bringen.

### Anmerkung

Für verschiedene Größen von verglichenen Arrays und mit dem angegebenen Wert count = WHOLE\_ARRAY für den Fall, wenn ein Array ist eine exakte Teilmenge der anderen, gibt die Funktion 0 nicht (Arrays werden gleich nicht.) In diesem Fall wird das Ergebnis von Vergleichen der Größe dieser Arrays: -1, wenn die Größe der array1 [] weniger als die Größe des array2 [] oder 1 andernfalls.

## ArrayFree

Befreit Puffer jedes dynamischen Arrays und stellt die Größe der Null-Dimension auf 0 ein.

```
void ArrayFree(
    void& array[]    // Array
);
```

### Parameter

`array[]`  
 [in] Dynamisches Array.

### Rückgabewert

keinen Rückgabewert.

### Anmerkung

Bei der Schreibung des Skripts und der Indikatoren können eine Notwendigkeit im Einführung der `ArrayFree()` Funktion nicht so oft entstehen, denn am Ende des Skripts der gesamte verwendete Speicher sofort befreit wird und in den benutzereigenen Indikatoren stellt die Grundarbeit mit den Arrays den Zugriff auf die Indikator Puffers, deren Größe wird von der bearbeiten Subsystem des Terminals automatisch schaffen.

Wenn das Programm braucht, um Speicher in den komplizierten dynamischen Bedingungen zu verwalten, so wird die `ArrayFree ()` Funktion offensichtlich und sofort selbständig der Speicher zu befreien, der mit dem schon nutzlos dynamischen Array beschäftigt ist.

### Beispiel:

```
#include <Controls\Dialog.mqh>
#include <Controls\Button.mqh>
#include <Controls\Label.mqh>
#include <Controls\ComboBox.mqh>
// --- vorbestimmte Konstanten
#define X_START 0
#define Y_START 0
#define X_SIZE 280
#define Y_SIZE 300
//+-----+
//| Die Klasse des Dialoges für Arbeit mit dem Speichern |
//+-----+
class CMemoryControl : public CAppDialog
{
private:
    //--- die Größe des Arrays
    int      m_arr_size;
    // --- Arrays
    char      m_arr_char[];
    int      m_arr_int[];
    float     m_arr_float[];
```

```

double          m_arr_double[];
long            m_arr_long[];
//--- Beschriftungen
CLabel          m_lbl_memory_physical;
CLabel          m_lbl_memory_total;
CLabel          m_lbl_memory_available;
CLabel          m_lbl_memory_used;
CLabel          m_lbl_array_size;
CLabel          m_lbl_array_type;
CLabel          m_lbl_error;
CLabel          m_lbl_change_type;
CLabel          m_lbl_add_size;
//--- Schaltflächen
CButton         m_button_add;
CButton         m_button_free;
// --- Listen
CComboBox       m_combo_box_step;
CComboBox       m_combo_box_type;
// --- der aktuelle Wert des Array-Typ aus der Liste
int             m_combo_box_type_value;

public:
    CMemoryControl(void);
    ~CMemoryControl(void);

//--- Methode der Erzeugung eines Objekts der Klasse
virtual bool     Create(const long chart,const string name,const int subwin,const
//--- der Bearbeiter des Ereignisses von Chart
virtual bool     OnEvent(const int id,const long &lparam,const double &dparam,const

protected:
    //--- erstellen Sie der Beschriftung
    bool          CreateLabel(CLabel &lbl,const string name,const int x,const int y
    //--- erstellen Sie die Steuerelemente
    bool          CreateButton(CButton &button,const string name,const int x,const
    bool          CreateComboBoxStep(void);
    bool          CreateComboBoxType(void);
    //--- der Bearbeiter des Ereignisses
    void          OnClickButtonAdd(void);
    void          OnClickButtonFree(void);
    void          OnChangeComboBoxType(void);
    //--- die Methoden der Arbeit mit dem aktuellen Array
    void          CurrentArrayFree(void);
    bool          CurrentArrayAdd(void);
};
//+-----+
//| Die Befreiung den Speichern des aktuellen Arrays |
//+-----+
void CMemoryControl::CurrentArrayFree(void)
{

```

```

//--- Rückstellung der Größe des Arrays
m_arr_size=0;
//--- die Befreiung des Arrays
if(m_combo_box_type_value==0)
    ArrayFree(m_arr_char);
if(m_combo_box_type_value==1)
    ArrayFree(m_arr_int);
if(m_combo_box_type_value==2)
    ArrayFree(m_arr_float);
if(m_combo_box_type_value==3)
    ArrayFree(m_arr_double);
if(m_combo_box_type_value==4)
    ArrayFree(m_arr_long);
; }
//+-----+
//| Die Unternehmung den Speichern für den aktuelle Arrays hinzufügen|
//+-----+
bool CMemoryControl::CurrentArrayAdd(void)
{
//--- wenn die Größe der verwendeten Speicher ist mehr als die Größe des physikalische
if(TerminalInfoInteger(TERMINAL_MEMORY_PHYSICAL)/TerminalInfoInteger(TERMINAL_MEMORY_FREE)>m_arr_size)
    return(false);
//--- die Unternehmung den Speichern abhängig von dem aktuellen Typ zu wählen
if(m_combo_box_type_value==0 && ArrayResize(m_arr_char,m_arr_size)==-1)
    return(false);
if(m_combo_box_type_value==1 && ArrayResize(m_arr_int,m_arr_size)==-1)
    return(false);
if(m_combo_box_type_value==2 && ArrayResize(m_arr_float,m_arr_size)==-1)
    return(false);
if(m_combo_box_type_value==3 && ArrayResize(m_arr_double,m_arr_size)==-1)
    return(false);
if(m_combo_box_type_value==4 && ArrayResize(m_arr_long,m_arr_size)==-1)
    return(false);
//--- der Speicher ist gewählt
return(true);
; }
//+-----+
//| Der Bearbeitung des Ereignisses |
//+-----+
EVENT_MAP_BEGIN(CMemoryControl)
ON_EVENT(ON_CLICK,m_button_add,OnClickButtonAdd)
ON_EVENT(ON_CLICK,m_button_free,OnClickButtonFree)
ON_EVENT(ON_CHANGE,m_combo_box_type,OnChangeComboBoxType)
EVENT_MAP_END(CAppDialog)
//+-----+
//| Konstrukteur |
//+-----+
CMemoryControl::CMemoryControl(void)
{

```

```

};
//+-----+
//| Destruktur |
//+-----+
CMemoryControl::~CMemoryControl(void)
{
};
//+-----+
//| Methode der Erzeugung eines Objekts der Klasse |
//+-----+
bool CMemoryControl::Create(const long chart,const string name,const int subwin,
                           const int x1,const int y1,const int x2,const int y2)
{
//--- der Erzeugung eines Objekts der grundlegende Klasse
if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
    return(false);
//--- die Vorbereitung die Zeilen für die Beschriftungen
string str_physical="Memory physical = "+(string)TerminalInfoInteger(TERMINAL_MEMORY_PHYSICAL);
string str_total="Memory total = "+(string)TerminalInfoInteger(TERMINAL_MEMORY_TOTAL);
string str_available="Memory available = "+(string)TerminalInfoInteger(TERMINAL_MEMORY_AVAILABLE);
string str_used="Memory used = "+(string)TerminalInfoInteger(TERMINAL_MEMORY_USED);
//--- erstellen Sie die Beschriftungen
if(!CreateLabel(m_lbl_memory_physical,"physical_label",X_START+10,Y_START+5,str_physical,12,clrBlack))
    return(false);
if(!CreateLabel(m_lbl_memory_total,"total_label",X_START+10,Y_START+30,str_total,12,clrBlack))
    return(false);
if(!CreateLabel(m_lbl_memory_available,"available_label",X_START+10,Y_START+55,str_available,12,clrBlack))
    return(false);
if(!CreateLabel(m_lbl_memory_used,"used_label",X_START+10,Y_START+80,str_used,12,clrBlack))
    return(false);
if(!CreateLabel(m_lbl_array_type,"type_label",X_START+10,Y_START+105,"Array type = ",12,clrBlack))
    return(false);
if(!CreateLabel(m_lbl_array_size,"size_label",X_START+10,Y_START+130,"Array size = ",12,clrBlack))
    return(false);
if(!CreateLabel(m_lbl_error,"error_label",X_START+10,Y_START+155,"",12,clrRed))
    return(false);
if(!CreateLabel(m_lbl_change_type,"change_type_label",X_START+10,Y_START+185,"Change type",12,clrBlack))
    return(false);
if(!CreateLabel(m_lbl_add_size,"add_size_label",X_START+10,Y_START+210,"Add to array",12,clrBlack))
    return(false);
//--- erstellen Sie das Steuerelement
if(!CreateButton(m_button_add,"add_button",X_START+15,Y_START+245,"Add",12,clrBlue))
    return(false);
if(!CreateButton(m_button_free,"free_button",X_START+75,Y_START+245,"Free",12,clrBlue))
    return(false);
if(!CreateComboBoxType())
    return(false);
if(!CreateComboBoxStep())
    return(false);

```

```

//--- die Initialisierung der Variablen
    m_arr_size=0;
//--- die erfolgreiche Ausführung
    return(true);
; }
//+-----+
//| Erstellen Sie die Schaltfläche |
//+-----+
bool CMemoryControl::CreateButton(CButton &button,const string name,const int x,
                                const int y,const string str,const int font_size,
                                const int clr)
{
//--- erstellen Sie die Schaltfläche
    if(!button.Create(m_chart_id,name,m_subwin,x,y,x+50,y+20))
        return(false);
//--- der Text
    if(!button.Text(str))
        return(false);
//--- der Schriftgröße
    if(!button.FontSize(font_size))
        return(false);
//--- die Farbe der Beschriftung
    if(!button.Color(clr))
        return(false);
//--- fügen Sie die Schaltfläche in den Kontrollelementen hinzu
    if(!Add(button))
        return(false);
//--- die erfolgreiche Ausführung
    return(true);
; }
//+-----+
//| Erstellen Sie die Liste für die Größe des Arrays |
//+-----+
bool CMemoryControl::CreateComboBoxStep(void)
{
//--- erstellen Sie die Liste
    if(!m_combo_box_step.Create(m_chart_id,"step_combobox",m_subwin,X_START+100,Y_START)
        return(false);
//--- fügen Sie die Elemente in die Liste hinzu
    if(!m_combo_box_step.ItemAdd("100 000",100000))
        return(false);
    if(!m_combo_box_step.ItemAdd("1 000 000",1000000))
        return(false);
    if(!m_combo_box_step.ItemAdd("10 000 000",10000000))
        return(false);
    if(!m_combo_box_step.ItemAdd("100 000 000",100000000))
        return(false);
//--- stellen Sie das aktuelle Element der Liste ein
    if(!m_combo_box_step.SelectByValue(1000000))

```

```

        return(false);
//--- fügen Sie die Liste in die Kontrollelemente hinzu
        if(!Add(m_combo_box_step))
            return(false);
//--- die erfolgreiche Ausführung
        return(true);
    };
//+-----+
//| Erstellen Sie die Liste für den Typ des Arrays |
//+-----+
bool CMemoryControl::CreateComboBoxType(void)
{
//--- erstellen Sie die Liste
    if(!m_combo_box_type.Create(m_chart_id,"type_combobox",m_subwin,X_START+100,Y_START))
        return(false);
//--- fügen Sie die Elemente in die Liste hinzu
    if(!m_combo_box_type.ItemAdd("char",0))
        return(false);
    if(!m_combo_box_type.ItemAdd("int",1))
        return(false);
    if(!m_combo_box_type.ItemAdd("float",2))
        return(false);
    if(!m_combo_box_type.ItemAdd("double",3))
        return(false);
    if(!m_combo_box_type.ItemAdd("long",4))
        return(false);
//--- stellen Sie das aktuelle Element der Liste ein
    if(!m_combo_box_type.SelectByValue(3))
        return(false);
//--- behalten Sie das aktuelle Element der Liste
    m_combo_box_type_value=3;
//--- fügen Sie die Liste in die Kontrollelemente hinzu
    if(!Add(m_combo_box_type))
        return(false);
//--- die erfolgreiche Ausführung
    return(true);
};
//+-----+
//| Erstellen Sie der Beschriftung |
//+-----+
bool CMemoryControl::CreateLabel(CLabel &lbl,const string name,const int x,
                                const int y,const string str,const int font_size,
                                const int clr)
{
//--- erstellen Sie der Beschriftung
    if(!lbl.Create(m_chart_id,name,m_subwin,x,y,0,0))
        return(false);
//--- der Text
    if(!lbl.Text(str))

```

```

        return(false);
//--- der Schriftgröße
        if(!lbl.FontSize(font_size))
            return(false);
//--- die Farbe
        if(!lbl.Color(clr))
            return(false);
//--- fügen Sie der Beschriftung in die Kontrollelemente hinzu
        if(!Add(lbl))
            return(false);
//--- erfolgreich sein
        return(true);
    };
//+-----+
//| Der Bearbeiter des Ereignisses auf die "Add" Schaltfläche zu |
//| klicken |
//+-----+
void CMemoryControl::OnClickButtonAdd(void)
{
//--- erhöhen Sie die Größe des Arrays
    m_arr_size+=(int)m_combo_box_step.Value();
//--- versuchen Sie der Speicher für den aktuellen Array zu wählen
    if(CurrentArrayAdd())
    {
        //--- der Speicher ist gewählt, führen wir den aktuellen Zustand an dem Chart he
        m_lbl_memory_available.Text("Memory available = "+(string)TerminalInfoInteger(TERM
        m_lbl_memory_used.Text("Memory used = "+(string)TerminalInfoInteger(TERMINAL_MEM
        m_lbl_array_size.Text("Array size = "+IntegerToString(m_arr_size));
        m_lbl_error.Text("");
    };
    else
    {
        //--- es misslang, der Speicher zu wählen, führen Sie die Nachtrichtung vom Fehl
        m_lbl_error.Text("Array is too large, error!");
        //--- geben Sie die vorherige Größe des Arrays zurück
        m_arr_size-=(int)m_combo_box_step.Value();
    };
};
//+-----+
//| Der Bearbeiter des Ereignisses auf die "Free" Schaltfläche zu |
//| klicken |
//+-----+
void CMemoryControl::OnClickButtonFree(void)
{
//--- befreien Sie den Speichern des aktuellen Arrays
    CurrentArrayFree();
//--- führen wir den aktuellen Zustand an dem Chart heraus
    m_lbl_memory_available.Text("Memory available = "+(string)TerminalInfoInteger(TERM
    m_lbl_memory_used.Text("Memory used = "+(string)TerminalInfoInteger(TERMINAL_MEMOR

```



```

    m_lbl_array_size.Text("Array size = 0");
    m_lbl_error.Text("");
; }

//+-----+
//| Der Bearbeiter des Ereignisses der Änderungsliste |
//+-----+
void CMemoryControl::OnChangeComboBoxType(void)
{
    //--- prüfen Sie, ob sich der Typ von Array geändert hat
    if(m_combo_box_type.Value()!=m_combo_box_type_value)
    {
        //--- befreien Sie den Speicher des aktuellen Arrays
        OnClickButtonFree();
        //--- arbeiten sie mit dem anderen Typ von Array
        m_combo_box_type_value=(int)m_combo_box_type.Value();
        //--- führen wir den neuen Typ des Arrays an dem Chart heraus
        if(m_combo_box_type_value==0)
            m_lbl_array_type.Text("Array type = char");
        if(m_combo_box_type_value==1)
            m_lbl_array_type.Text("Array type = int");
        if(m_combo_box_type_value==2)
            m_lbl_array_type.Text("Array type = float");
        if(m_combo_box_type_value==3)
            m_lbl_array_type.Text("Array type = double");
        if(m_combo_box_type_value==4)
            m_lbl_array_type.Text("Array type = long");
    }
; }

//--- das Objekt der Klasse von CMemoryControl
CMemoryControl ExtDialog;

//+-----+
//| Expert initialization function |
//+-----+
int OnInit()
{
    //--- erstellen Sie das Dialog
    if(!ExtDialog.Create(0,"MemoryControl",0,X_START,Y_START,X_SIZE,Y_SIZE))
        return(INIT_FAILED);
    //--- der Start
    ExtDialog.Run();
    //---
    return(INIT_SUCCEEDED);
; }

//+-----+
//| Expert deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
    //---

```

```
ExtDialog.Destroy(reason);  
};  
//+-----+  
//| Expert chart event function |  
//+-----+  
void OnChartEvent(const int id,  
                  const long &lparam,  
                  const double &dparam,  
                  const string &sparam)  
{  
    ExtDialog.ChartEvent(id, lparam, dparam, sparam);  
};
```

## ArrayGetAsSeries

Prüft Richtung des Arrayindizierens.

```
bool ArrayGetAsSeries(  
    const void& array[]    // Array für Prüfung  
);
```

### Parameter

*array*

[in] Das geprüfte Array.

### Rückgabewert

Gibt [true](#) zurück, wenn die Flagge AS\_SERIES für das angegebene Array eingestellt wird, d.h. der Arrayzugang wird umgekehrt wie in einer Zeitreihe durchgeführt. [Zeitreihe](#) unterscheidet sich vom normalen Array dadurch, dass Indizieren der Elemente der Zeitreihe vom Arrayende zum Arrayanfang durchgeführt (von den neuesten Daten zu den ältesten).

### Bemerkung

Für Prüfung des Arrays auf Zugehörigkeit muss die Funktion [ArrayIsSeries\(\)](#) verwendet werden. Arrays der Preisdaten, die als Eingabeparameter in die Funktion [OnCalculate\(\)](#) übertragen werden, nicht haben Richtung des Indizierens wie in Zeitreihen. Die notwendige Richtung des Indizierens kann durch die Funktion [ArraySetAsSeries\(\)](#) eingestellt werden.

### Beispiel:

```

#property description "der Indikator rechnet die absoluten Werten des Unterschieds zw
#property description "Open und Close oder High und Low, und zeichnet sie im abgesonde
#property description "in Form vom Histogramm."
//--- der Indikator Einstellungen
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots 1
//---- plot
#property indicator_type1 DRAW_HISTOGRAM
#property indicator_style1 STYLE_SOLID
#property indicator_width1 3
//--- Eingabeparameters
input bool InpAsSeries=true; // Die Richtung des Indizierens in der Indikator Puffer
input bool InpPrices=true; // Preise für die Berechnung (true - Open,Close; false -
//--- Indikator Puffer
double ExtBuffer[];
//+-----+
//| Die Berechnung der Indikatorwerte |
//+-----+
void CandleSizeOnBuffer(const int rates_total,const int prev_calculated,
                        const double &first[],const double &second[],double &buffer[])
{
//---Variable zur Berechnung den Anfang des Bars
int start=prev_calculated;
//--- wenn die Indikatorwerte auf vorhergehend Tick schon berechnet waren, so ist auf c
if(prev_calculated>0)
start--;
//--- bestimmen Sie die Richtung der Indizierung in Arrays
bool as_series_first=ArrayGetAsSeries(first);
bool as_series_second=ArrayGetAsSeries(second);
bool as_series_buffer=ArrayGetAsSeries(buffer);
//--- ändern Sie die Richtung der Indizierung auf die direkte, wenn nötig
if(as_series_first)
ArraySetAsSeries(first,false);
if(as_series_second)
ArraySetAsSeries(second,false);
if(as_series_buffer)
ArraySetAsSeries(buffer,false);
//--- berechnen Sie die Indikatorwerte
for(int i=start;i<rates_total;i++)
buffer[i]=MathAbs(first[i]-second[i]);
}
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- Bindung von Indikator Puffers
SetIndexBuffer(0,ExtBuffer);
//--- stellen Sie die Richtung der Indizierung in dem Indikator Puffer ein
ArraySetAsSeries(ExtBuffer,InpAsSeries);
//--- prüfen wir für welche Preise es wird der Indikator berechnet
if(InpPrices)
{
//--- Open und Close Preise
PlotIndexSetString(0,PLOT_LABEL,"BodySize");
//--- erstellen Sie die Farbe des Indikators
PlotIndexSetInteger(0,PLOT_LINE_COLOR,clrOrange);
}
else
{

```

```

    //--- High und Low Preise
    PlotIndexSetString(0,PLOT_LABEL,"ShadowSize");
    //--- erstellen Sie die Farbe des Indikators
    PlotIndexSetInteger(0,PLOT_LINE_COLOR,clrDodgerBlue);
}
//---
    return(INIT_SUCCEEDED);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
    //--- die Berechnung des Indikators je nach der Wert der Fahne
    if(InpPrices)
        CandleSizeOnBuffer(rates_total,prev_calculated,open,close,ExtBuffer);
    else
        CandleSizeOnBuffer(rates_total,prev_calculated,high,low,ExtBuffer);
    //--- den Wert prev_calculated für den nächsten Anruf zurückgeben
    return(rates_total);
}

```

Sehen Sie auch

[Zugang zu Zeitreihen](#), [ArraySetAsSeries](#)

## ArrayInitialize

Initialisiert das numerische Array durch den vorgegebenen Wert.

### For initialization of an array of char type

```
int ArrayInitialize(  
    char    array[],    // initialisiertes array  
    char    value       // der zu gestellte wert  
);
```

### For initialization of an array of short type

```
int ArrayInitialize(  
    short   array[],    // initialisiertes array  
    short   value       // der zu gestellte wert  
);
```

### For initialization of an array of int type

```
int ArrayInitialize(  
    int     array[],    // initialisiertes array  
    int     value       // der zu gestellte wert  
);
```

### For initialization of an array of long type

```
int ArrayInitialize(  
    long    array[],    // initialisiertes array  
    long    value       // der zu gestellte wert  
);
```

### For initialization of an array of float type

```
int ArrayInitialize(  
    float   array[],    // initialisiertes array  
    float   value       // der zu gestellte wert  
);
```

### For initialization of an array of double type

```
int ArrayInitialize(  
    double  array[],    // initialisiertes array  
    double  value       // der zu gestellte wert  
);
```

### For initialization of an array of bool type

```
int ArrayInitialize(  
    bool    array[],    // initialisiertes array  
    bool    value       // der zu gestellte wert  
);
```

## For initialization of an array of uint type

```
int ArrayInitialize(
    uint    array[],    // initialisiertes array
    uint    value       // der zu gestellte wert
);
```

## Parameter

*array[]*

[out] Das numerische Array, das initialisiert werden muss.

*value*

[in] Neuer Wert, der für alle Elemente des Arrays eingestellt werden muss.

## Rückgabewert

Anzahl der Elemente.

## Bemerkung

Die Funktion [ArrayResize\(\)](#) erlaubt die Größe für ein Array mit einem Reserve vorzugeben, damit es größer wird ohne physische Umverteilung des Speichers. Das wird für Operationsgeschwindigkeit gemacht, denn Operationen der Speicherverteilung sind langsam genug.

Die Initialisierung des Arrays vom Ausdruck [ArrayInitialize\(array, init\\_val\)](#) bedeutet nicht die Initialisierung durch denselben Wert der Elemente der Reserve, die für dieses Array gewählt ist. Bei weiteren Vergrößerungen des Arrays *array* durch die Funktion [ArrayResize\(\)](#) innerhalb des laufenden Reserves, am Ende des Arrays werden die Elemente eingefügt, deren Werte nicht bestimmt sind und am öftesten sind sie nicht *init\_val* gleich.

## Beispiel:

```
void OnStart()
{
    //--- dynamisches Array
    double array[];
    //--- geben wir die Größe des Arrays für 100 Elemente vor und reservieren wir noch ein
    ArrayResize(array,100,10);
    //--- initialisieren wir die Elemente des Arrays durch den Wert EMPTY_VALUE=DBL_MAX
    ArrayInitialize(array,EMPTY_VALUE);
    Print("Werte der letzten 10 Elemente des Arrays nach der Initialisierung");
    for(int i=90;i<100;i++) printf("array[%d]=%G",i,array[i]);
    //--- vergrößern wir das Array um 5 Elemente
    ArrayResize(array,105);
    Print("Werte der letzten 10 Elemente des Arrays nach ArrayResize(array,105)");
    //--- Werte der letzten 5 Elemente wurden aus Reservepuffer erhalten
    for(int i=95;i<105;i++) printf("array[%d] = %G",i,array[i]);
}
```

## ArrayFill

Füllt das numerische Array mit dem angegebenen Wert.

```
void ArrayFill(
    void& array[],      // Array
    int start,          // Index des Anfangselements
    int count,          // Anzahl der Elemente
    void value          // Wert, der mit Array gefüllt wird
);
```

### Parameter

*array[]*

[out] Array des einfachen Typs ([char](#), [uchar](#), [short](#), [ushort](#), [int](#), [uint](#), [long](#), [ulong](#), [bool](#), [color](#), [datetime](#), [float](#), [double](#)).

*start*

[in] Index des Anfangselements (von welchem Element auszufüllen). Anbei wird die eingestellte [Seriengröße Flagge](#) nicht berücksichtigt.

*count*

[in] Die Anzahl der Elemente, die gefüllt werden muss.

*value*

[in] Wert, der mit Array gefüllt wird.

### Rückgabewert

Keinen Rückgabewert.

### Anmerkung

Beim Aufruf der ArrayFill() Funktion immer wird die gewöhnliche Richtung der Indexierung - von links nach rechts verstanden, d.h. die Veränderung den Zugang auf die Elemente des Arrays mit Hilfe der [ArraySetAsSeries\(\)](#) Funktion wird nicht beachtet.

Das mehrdimensionale Array bei Bearbeitung von der Funktion ArrayFill werden eindimensional, zum Beispiel, wird das Array array[2][4] wie array[8], deshalb bei der Arbeit mit diesem Array zulässig vorgestellt, den Index des Anfangselementes gleich 5 zu bezeichnen. So wird der Aufruf von ArrayFill(array, 5, 2, 3.14) für das Array array[2][4] der Wert die 3.14 Elemente des Arrays array[1][1] und array[1][2] ausfüllen.

### Beispiel:

```
void OnStart()
{
    //--- deklarieren wir ein dynamisches Array
    int a[];
    //--- stellen wir die Größe ein
    ArrayResize(a,10);
    //--- füllen wir die ersten 5 Elementen mit dem Wert 123
    ArrayFill(a,0,5,123);
    //--- füllen wir 5 Elementen (seit 5.) mit dem Wert 456
    ArrayFill(a,5,5,456);
}
```



```
//--- Die Werte aller Elemente ausgeben
for(int i=0;i<ArraySize(a);i++) printf("a[%d] = %d",i,a[i]);
}
```

## ArrayIsDynamic

Prüft, ob das Array dynamisch ist.

```
bool ArrayIsDynamic(  
    const void& array[]    // das geprüfte Array  
);
```

### Parameter

*array[]*

[in] Das geprüfte Array.

### Rückgabewert

Gibt true zurück, wenn das angegebene Array dynamisch ist, anderenfalls false.

Beispiel:

```

#property description "Dieser Indikator berechnet die Werte nicht, und einmal versucht
#property description "Der Anruf der Funktion ArrayFree() zu drei Arrays: dynamisch, s
#property description "dem Indikator Puffer. Die Ergebnisse werden in die Zeitschrift
//--- der Indikator Einstellungen
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots 1
//--- globale Variablen
double ExtDynamic[]; // das dynamische Array
double ExtStatic[100]; // das statische Array
bool ExtFlag=true; // die Flagge
double ExtBuff[]; // der Indikator Puffer
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- trennen Sie den Speicher für den Array ab
ArrayResize(ExtDynamic,100);
//--- Bindung von Array zum Indikator-Puffer
SetIndexBuffer(0,ExtBuff);
PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
return(INIT_SUCCEEDED);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const int begin,
               const double &price[])
{
//--- führen Sie die Analyse einmal durch
if(ExtFlag)
{
//--- versuchen Sie den Speicher für die Arrays zu befreien
//--- 1. Das dynamische Array
Print("+++++++");
Print("1. Überprüfung des dynamischen Arrays:");
Print("Die Größe bis zur Befreiung des Speichers = ",ArraySize(ExtDynamic));
Print("Es ist das dynamische Array = ",ArrayIsDynamic(ExtDynamic) ? "Ja" : "Nein");
//--- versuchen Sie den Speicher des Arrays zu befreien
ArrayFree(ExtDynamic);
Print("Die Größe nach der Befreiung des Speichers = ",ArraySize(ExtDynamic));
//--- 2. Das statische Array
Print("2. Überprüfung des statischen Arrays:");
Print("Die Größe bis zur Befreiung des Speichers = ",ArraySize(ExtStatic));
Print("Es ist das dynamische Array = ",ArrayIsDynamic(ExtStatic) ? "Ja" : "Nein");
//--- versuchen Sie den Speicher des Arrays zu befreien
ArrayFree(ExtStatic);
Print("Die Größe nach der Befreiung des Speichers = ",ArraySize(ExtStatic));
//--- 3. Indikator Puffer
Print("3. Überprüfung des Indikator Puffers:");
Print("Die Größe bis zur Befreiung des Speichers = ",ArraySize(ExtBuff));
Print("Es ist das dynamische Array = ",ArrayIsDynamic(ExtBuff) ? "Ja" : "Nein");
//--- versuchen Sie den Speicher des Arrays zu befreien
ArrayFree(ExtBuff);
Print("Die Größe nach der Befreiung des Speichers = ",ArraySize(ExtBuff));
//--- ändern Sie den Wert der Flagge
ExtFlag=false;
}
}

```

```
    }  
    ///--- den Wert prev_calculated für den nächsten Anruf zurückgeben  
    return(rates_total);  
}
```

Sehen Sie auch

[Zugang zu Zeitreihen und Indikatoren](#)

## ArrayIsSeries

Prüft, ob das Array Zeitreihe ist.

```
bool ArrayIsSeries(  
    const void& array[]    // das geprüfte Array  
);
```

### Parameter

*array[]*

[in] Das geprüfte Array.

### Rückgabewert

Gibt true zurück, wenn das geprüfte Array ein Array-Zeitreihe ist, anderenfalls false. Arrays, die als Parameter der Funktion [OnCalculate\(\)](#) übertragen werden, müssen auf Zugangsordnung zu Elementen des Arrays durch die Funktion [ArrayGetAsSeries\(\)](#) geprüft werden.

**Beispiel:**

```

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots 1
//---- plot Label1
#property indicator_label1 "Label1"
#property indicator_type1 DRAW_LINE
#property indicator_color1 clrRed
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//--- Indikator Puffer
double Label1Buffer[];
//+-----+
//| Custom indicator initialization function |
//+-----+
void OnInit()
{
//--- Bindung von Array zum Indikator-Puffer
SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
//---
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
//---
if(ArrayIsSeries(open))
    Print("open[] is timeseries");
else
    Print("open[] is not timeseries!!!");
//--- den Wert prev_calculated für den nächsten Anruf zurückgeben
return(rates_total);
}

```

Sehen Sie auch

[Zugang zu Zeitreihen und Indikatoren](#)

## ArrayMaximum

Sucht im eindimensionalen numerischen Array das maximale Element.

```
int ArrayMaximum(
    const void& array[],           // Array für die Suche
    int start=0,                  // Anfangsindex für die Suche
    int count=WHOLE_ARRAY         // Anzahl der geprüften Elemente
);
```

### Parameter

`array[]`

[in] Numerisches Array, in dem Suche durchgeführt wird.

`start=0`

[in] Anfangsindex für die Suche.

`count=WHOLE_ARRAY`

[in] Anzahl der Elementen für die Suche. Als Voreinstellung wird das ganze Array kopiert (count=WHOLE\_ARRAY).

### Rückgabewert

Funktion gibt Index des gefundenen Elementes unter Beachtung von Seriengröße eines Arrays. Beim Misserfolg gibt die Funktion -1 zurück.

### Beispiel:

```
#property description "Der Indikator zeigt die Kerzen mit dem älteren auf die aktuelle
//--- Indikator Einstellungen
#property indicator_chart_window
#property indicator_buffers 16
#property indicator_plots 8
//---- plot 1
#property indicator_label1 "BearBody"
#property indicator_color1 clrSeaGreen,clrSeaGreen
//---- plot 2
#property indicator_label2 "BearBodyEnd"
#property indicator_color2 clrSeaGreen,clrSeaGreen
//---- plot 3
#property indicator_label3 "BearShadow"
#property indicator_color3 clrSalmon,clrSalmon
//---- plot 4
#property indicator_label4 "BearShadowEnd"
#property indicator_color4 clrSalmon,clrSalmon
//---- plot 5
#property indicator_label5 "BullBody"
#property indicator_color5 clrOlive,clrOlive
//---- plot 6
#property indicator_label6 "BullBodyEnd"
#property indicator_color6 clrOlive,clrOlive
```

```

//---- plot 7
#property indicator_label7 "BullShadow"
#property indicator_color7 clrSkyBlue,clrSkyBlue
//---- plot 8
#property indicator_label8 "BullShadowEnd"
#property indicator_color8 clrSkyBlue,clrSkyBlue
//--- vordefinierte Konstante
#define INDICATOR_EMPTY_VALUE 0.0
//--- Eingabeparameters
input ENUM_TIMEFRAMES InpPeriod=PERIOD_H4; // Der Zeitrahmen für die Berechnung
input datetime InpDateStart=D'2013.01.01 00:00'; // das Anfangsdatum der Analyse
//--- die Indikator Puffers für die Bären Kerzen
double ExtBearBodyFirst[];
double ExtBearBodySecond[];
double ExtBearBodyEndFirst[];
double ExtBearBodyEndSecond[];
double ExtBearShadowFirst[];
double ExtBearShadowSecond[];
double ExtBearShadowEndFirst[];
double ExtBearShadowEndSecond[];
//--- die Indikator Puffers für die Bullen Kerzen
double ExtBullBodyFirst[];
double ExtBullBodySecond[];
double ExtBullBodyEndFirst[];
double ExtBullBodyEndSecond[];
double ExtBullShadowFirst[];
double ExtBullShadowSecond[];
double ExtBullShadowEndFirst[];
double ExtBullShadowEndSecond[];
//--- globale Variablen
datetime ExtTimeBuff[]; // Puffer für Zeit mit der höheren Zeitrahmen
int ExtSize=0; // die Größe des Zeitpuffers
int ExtCount=0; // der Index innerhalb des Zeitpuffers
int ExtStartPos=0; // die Anfangsposition für die Berechnung des Indikators
bool ExtStartFlag=true; // Die Hilfsflagge für das Erhalten der Anfangsposition
datetime ExtCurrentTime[1]; // die letzte Zeit wenn der Bar auf dem älteren Zeitrahmen
datetime ExtLastTime; // die letzte Zeit auf dem älteren Zeitrahmen, für den die Berechnung
bool ExtBearFlag=true; // die Flagge für die Bestimmung der Ordnung der Speicherung
bool ExtBullFlag=true; // die Flagge für die Bestimmung der Ordnung der Speicherung
int ExtIndexMax=0; // der Index des maximalen Elementes in dem Array
int ExtIndexMin=0; // der Index des minimalen Elementes in dem Array
int ExtDirectionFlag=0; // die Richtung Die Bewegungen des Preises in der aktuellen Zeit
//--- der Einzug zwischen dem Eröffnungspreis und Schlusspreis der Kerze für die richtige
const double ExtEmptyBodySize=0.2*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//+-----+
//| Zeichnung der Hauptteil der Kerze |
//+-----+
void FillCandleMain(const double &open[],const double &close[],
const double &high[],const double &low[],

```



```

        const int start,const int last,const int fill_index,
        int &index_max,int &index_min)
    {
        //--- finden Sie die Indexe der maximalen und minimalen Elemente in den Arrays
        index_max=ArrayMaximum(high,ExtStartPos,last-start+1); // Maximum in Hoch
        index_min=ArrayMinimum(low,ExtStartPos,last-start+1); // Minimum in Tief
        //--- bestimmen Sie, wie viel Bars auf den aktuellen Zeitrahmen wir werden zeichnen
        int count=fill_index-start+1;
        //--- wenn der Schlusspreis am ersten Bar mehr als der Schlusspreis am letzten Bar - es
        if(open[start]>close[last])
        {
            //--- wenn früher die Kerze Bullen war, so freimachen wir die Werte den Bären Indikat
            if(ExtDirectionFlag!=-1)
                ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowSecond);
            //--- Bären Kerze
            ExtDirectionFlag=-1;
            //--- bilden Sie die Kerze
            FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShadowSecond,
                close[last],high[index_max],low[index_min],start,count,ExtBearFlag);
            //--- Funktionsausgang
            return;
        }
        //--- wenn der Schlusspreis am ersten Bar weniger als der Schlusspreis am letzten - es
        if(open[start]<close[last])
        {
            //--- wenn früher die Kerze Bären war, so freimachen wir die Werte den Bären Indikat
            if(ExtDirectionFlag!=1)
                ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShadowSecond);
            //--- Bullen Kerze
            ExtDirectionFlag=1;
            //--- bilden Sie die Kerze
            FormCandleMain(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowSecond,
                open[start],high[index_max],low[index_min],start,count,ExtBullFlag);
            //--- Funktionsausgang
            return;
        }
        //--- wenn sie in diesem Teil der Funktion, so bedeutet es, so stellt sich der Eröffnung
        //--- dem Schlusspreis am letzten Bar; wir werden solche Kerze Bären halten
        //--- wenn früher die Kerze Bullen war, so freimachen wir die Werte der Bullen Indikat
        if(ExtDirectionFlag!=-1)
            ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowSecond);
        //--- Bären Kerze
        ExtDirectionFlag=-1;
        //--- wenn Schlusspreise und Eröffnungspreise gleich sind, dann verschoben Sie für die
        if(high[index_max]!=low[index_min])
            FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShadowSecond,
                open[start]-ExtEmptyBodySize,high[index_max],low[index_min],start,count,ExtBearFlag);
        else
            FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShadowSecond,
                open[start],high[index_max],low[index_min],start,count,ExtBearFlag);
    }

```

```

        open[start],open[start]-ExtEmptyBodySize,high[index_max],
        high[index_max]-ExtEmptyBodySize,start,count,ExtBearFlag);

; }
//+-----+
//| Zeichnung das Ende der Kerze |
//+-----+
void FillCandleEnd(const double &open[],const double &close[],
                  const double &high[],const double &low[],
                  const int start,const int last,const int fill_index,
                  const int index_max,const int index_min)
{
//--- wenn es nur eine Bar gibt, dann wir nicht zeichnen
    if(last-start==0)
        return;
//--- wenn der Schlusspreis am ersten Bar mehr als der Schlusspreis am letzten Bar - es
    if(open[start]>close[last])
    {
        //--- bilden Sie das Ende der Kerze
        FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                      open[start],close[last],high[index_max],low[index_min],fill_index,
        //--- Funktionsausgang
        return;
    }
; }
//--- wenn der Schlusspreis am ersten Bar weniger als der Schlusspreis am letzten - es
    if(open[start]<close[last])
    {
        //--- bilden Sie das Ende der Kerze
        FormCandleEnd(ExtBullBodyEndFirst,ExtBullBodyEndSecond,ExtBullShadowEndFirst,Ext
                      close[last],open[start],high[index_max],low[index_min],fill_index,
        //--- Funktionsausgang
        return;
    }
; }
//--- wenn sie in diesem Teil der Funktion, so bedeutet es, so stellt sich der Eröffn
//--- dem Schlusspreis am letzten Bar; wir werden solche Kerze Bären halten
//--- bilden Sie das Ende der Kerze
    if(high[index_max]!=low[index_min])
        FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                      open[start]-ExtEmptyBodySize,high[index_max],low[index_min],fill_i
    else
        FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                      open[start]-ExtEmptyBodySize,high[index_max],high[index_max]-ExtEn
; }
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- Überprüfen Sie die Indikator Periode
    if(!CheckPeriod((int)Period(),(int)InpPeriod))

```

```

        return(INIT_PARAMETERS_INCORRECT);
//--- die Anzeige der Preisdaten auf dem Vordergrund
    ChartSetInteger(0,CHART_FOREGROUND,0,1);
//--- Bindung von Indikator Puffers
    SetIndexBuffer(0,ExtBearBodyFirst);
    SetIndexBuffer(1,ExtBearBodySecond);
    SetIndexBuffer(2,ExtBearBodyEndFirst);
    SetIndexBuffer(3,ExtBearBodyEndSecond);
    SetIndexBuffer(4,ExtBearShadowFirst);
    SetIndexBuffer(5,ExtBearShadowSecond);
    SetIndexBuffer(6,ExtBearShadowEndFirst);
    SetIndexBuffer(7,ExtBearShadowEndSecond);
    SetIndexBuffer(8,ExtBullBodyFirst);
    SetIndexBuffer(9,ExtBullBodySecond);
    SetIndexBuffer(10,ExtBullBodyEndFirst);
    SetIndexBuffer(11,ExtBullBodyEndSecond);
    SetIndexBuffer(12,ExtBullShadowFirst);
    SetIndexBuffer(13,ExtBullShadowSecond);
    SetIndexBuffer(14,ExtBullShadowEndFirst);
    SetIndexBuffer(15,ExtBullShadowEndSecond);
//--- stellen Sie einige Werte der Eigenschaften für die Konstruktion des Indikators e
    for(int i=0;i<8;i++)
    {
        PlotIndexSetInteger(i,PLOT_DRAW_TYPE,DRAW_FILLING); // der Typ der graphischen F
        PlotIndexSetInteger(i,PLOT_LINE_STYLE,STYLE_SOLID); // der Stil der Linie auf d
        PlotIndexSetInteger(i,PLOT_LINE_WIDTH,1);           // die Dicke der Linie auf c
    };
//---
    return(INIT_SUCCEEDED);
};
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
//--- wenn es noch keine berechnete Bars gibt
    if(prev_calculated==0)
    {
        //--- bekommen Sie die Zeit wenn der Bar auf dem älteren Zeitrahmen erschienen v
        if(!GetTimeData())

```

```

        return(0);
    ;}

//--- stellen Sie die gerade Richtung der Indexierung ein
    ArraySetAsSeries(time,false);
    ArraySetAsSeries(high,false);
    ArraySetAsSeries(low,false);
    ArraySetAsSeries(open,false);
    ArraySetAsSeries(close,false);
//--- Variable zur Berechnung den Anfang des Bars
    int start=prev_calculated;
//--- wenn der Bar gebildet ist, so berechnen wir den Indikatorwert darauf
    if(start!=0 && start==rates_total)
        start--;
//--- die Zyklus von die Berechnung der Indikatorwerte
    for(int i=start;i<rates_total;i++)
    {
        //--- füllen Sie die i-ten Elemente des Indikator Puffers mit dem leeren Werten
        FillIndicatorBuffers(i);
        //--- führen Sie die Berechnung für die Bars seit dem Datum InpDateStart durch
        if(time[i]>=InpDateStart)
        {
            //--- zum ersten Mal bestimmen Sie eine Position mit der wir werden die Werte
            if(ExtStartFlag)
            {
                //--- speichern Sie der Index des Anfangsbar
                ExtStartPos=i;
                //--- bestimmen Sie das erste Datum auf dem älteren Zeitrahmen, das mehr a
                while(time[i]>=ExtTimeBuff[ExtCount])
                    if(ExtCount<ExtSize-1)
                        ExtCount++;
                //--- ändern Sie die Flaggewert, um in diesen Block nicht mehr zu geraten
                ExtStartFlag=false;
            }
            //--- prüfen Sie, ob es noch Elemente in das Array sind
            if(ExtCount<ExtSize)
            {
                //--- warten Sie, wenn der Zeitwert auf dem aktuellen Zeitrahmen der Wert
                if(time[i]>=ExtTimeBuff[ExtCount])
                {
                    //--- zeichnen Sie den Hauptteil der Kerze (ohne Zeichnung zwischen den
                    FillCandleMain(open,close,high,low,ExtStartPos,i-1,i-2,ExtIndexMax,ExtI
                    //--- zeichnen Sie das Ende der Kerze (das Gebiet zwischen der letzten
                    FillCandleEnd(open,close,high,low,ExtStartPos,i-1,i-1,ExtIndexMax,ExtI
                    //--- bewegen Sie die Anfangsposition für die Zeichnung der nächsten Ke
                    ExtStartPos=i;
                    //--- erhöhen Sie der Array Zähler
                    ExtCount++;
                }
            }
            else

```

```

        continue;
    ;}
else
{
    //--- der Fehlerwert zu stürzen
    ResetLastError();
    //--- bekommen Sie die letzte Datum auf dem älteren Zeitrahmen
    if(CopyTime(Symbol(), InpPeriod, 0, 1, ExtCurrentTime)==-1)
    {
        Print("Fehler beim Kopieren die Daten, Kode = ", GetLastError());
        return(0);
    }
    //--- wenn das neue Datum mehr ist, so beenden wir die Bildung der Kerze
    if(ExtCurrentTime[0]>ExtLastTime)
    {
        //--- freimachen Sie das Gebiet zwischen der letzten und vorletzten Bar
        ClearEndOfBodyMain(i-1);
        //--- zeichnen Sie dieses Gebiet mit dem Hilfs-Indikator Puffers
        FillCandleEnd(open, close, high, low, ExtStartPos, i-1, i-1, ExtIndexMax, ExtIndexMin);
        //--- bewegen Sie die Anfangsposition für die Zeichnung der nächsten Kerze
        ExtStartPos=i;
        //--- stürzen Sie die Fahne der Preisrichtung
        ExtDirectionFlag=0;
        //--- speichern Sie das neue letzte Datum
        ExtLastTime=ExtCurrentTime[0];
    }
    else
    {
        //--- bilden Sie die Kerze
        FillCandleMain(open, close, high, low, ExtStartPos, i, i, ExtIndexMax, ExtIndexMin);
    }
    ;}
; }

//--- den Wert prev_calculated für den nächsten Anruf zurückgeben
return(rates_total);
; }

//+-----+
//| Überprüfen Sie die Indikator Periode auf die Korrektheit |
//+-----+
bool CheckPeriod(int current_period, int high_period)
{
    //--- die Periode des Indikators soll mehr als der Zeitrahmen sein, auf dem er dargestellt wird
    if(current_period>=high_period)
    {
        Print("Fehler! Der Wert von der Indikator Periode soll mehr als der aktuellen Zeitrahmen sein");
        return(false);
    }
    ;}

//--- wenn die Periode des Indikators eine Woche oder einer Monat ist, so ist die Periode korrekt

```

```

    if(high_period>32768)
        return(true);
//--- bringen Sie die Werte der Perioden bis zu Minuten
    if(high_period>30)
        high_period=(high_period-16384)*60;
    if(current_period>30)
        current_period=(current_period-16384)*60;
//--- die Periode des Indikators soll Fach der Zeitrahmen sein, auf dem er dargestellt
    if(high_period%current_period!=0)
    {
        Print("Fehler! Der Wert der Periode des Indikators soll Fach der aktuellen Zeitrahmen sein");
        return(false);
    };
//--- die Periode des Indikators soll der Wert der Zeitrahmen übertreten, auf dem er dargestellt
    if(high_period/current_period<3)
    {
        Print("Fehler! Der Wert der Periode des Indikators soll der Wert der aktuellen Zeitrahmen sein");
        return(false);
    };
//--- die Periode des Indikators ist für den aktuellen Zeitrahmen korrekt
    return(true);
};
//+-----+
//| Bekommen Sie die Daten der Zeit auf dem älteren Zeitrahmen |
//+-----+
bool GetTimeData(void)
{
//--- der Fehlerwert zu stürzen
    ResetLastError();
//--- kopieren Sie alle Daten auf die aktuelle Zeit
    if(CopyTime(Symbol(),InpPeriod,InpDateStart,TimeCurrent(),ExtTimeBuff)==-1)
    {
        //--- der Fehlerwert zu stürzen
        int code=GetLastError();
        //--- drucken Sie der Fehlertext
        PrintFormat("Fehler beim Kopieren die Daten! %s",code==4401
            ? "Die Geschichte wird noch beladen!"
            : "Kode = "+IntegerToString(code));
        //--- geben Sie false für den nochmaligen Versuch der Beladung der Daten zurück
        return(false);
    };
//--- bekommen Sie die Größe des Arrays
    ExtSize=ArraySize(ExtTimeBuff);
//--- stellen Sie der Index des Zyklus für das Array gleich der Null ein
    ExtCount=0;
//--- stellen Sie die Position der aktuellen Kerze in diesem Zeitrahmen gleich der Null
    ExtStartPos=0;
    ExtStartFlag=true;
//--- bekommen Sie der letzten Wert der Zeit auf dem älteren Zeitrahmen

```

```

    ExtLastTime=ExtTimeBuff[ExtSize-1];
//--- die erfolgreiche Ausführung
    return(true);
};
//+-----+
//| Die Funktion bildet den Hauptteil der Kerze. Je nach dem Wert |
//| der Flagge, die Funktion bestimmt, welche Daten sich in welche |
//| Arrays für die korrekte Abbildung einschreiben sollen.          |
//+-----+
void FormCandleMain(double &body_fst[],double &body_snd[],
                    double &shadow_fst[],double &shadow_snd[],
                    const double fst_value,const double snd_value,
                    const double fst_extremum,const double snd_extremum,
                    const int start,const int count,const bool flag)
{
//--- prüfen Sie der Wert der Flagge
    if(flag)
    {
        //--- bilden Sie der Körper der Kerze
        FormMain(body_fst,body_snd,fst_value,snd_value,start,count);
        //--- bilden Sie der Schatten der Kerze
        FormMain(shadow_fst,shadow_snd,fst_extremum,snd_extremum,start,count);
    };
    else
    {
        //--- bilden Sie der Körper der Kerze
        FormMain(body_fst,body_snd,snd_value,fst_value,start,count);
        //--- bilden Sie der Schatten der Kerze
        FormMain(shadow_fst,shadow_snd,snd_extremum,fst_extremum,start,count);
    };
};
//+-----+
//| Die Funktion bildet das Ende der Kerze. Je nach dem Wert der |
//| Flagge, die Funktion bestimmt, welche Daten sich in welche Arrays|
//| für die korrekte Abbildung einschreiben sollen.                |
//+-----+
void FormCandleEnd(double &body_fst[],double &body_snd[],
                   double &shadow_fst[],double &shadow_snd[],
                   const double fst_value,const double snd_value,
                   const double fst_extremum,const double snd_extremum,
                   const int end,bool &flag)
{
//--- prüfen Sie der Wert der Flagge
    if(flag)
    {
        //--- bilden Sie das Ende der Körper der Kerze
        FormEnd(body_fst,body_snd,fst_value,snd_value,end);
        //--- bilden Sie das Ende der Schatten der Kerze
        FormEnd(shadow_fst,shadow_snd,fst_extremum,snd_extremum,end);
    };
};

```

```

    //--- ändern Sie der Wert der Flagge gegen das Entgegengesetzte
    flag=false;
};
else
{
    //--- bilden Sie das Ende der Körper der Kerze
    FormEnd(body_fst,body_snd,snd_value,fst_value,end);
    //--- bilden Sie das Ende der Schatten der Kerze
    FormEnd(shadow_fst,shadow_snd,snd_extremum,fst_extremum,end);
    //--- ändern Sie der Wert der Flagge gegen das Entgegengesetzte
    flag=true;
};
};

//+-----+
//| Die Entleerung das Ende der Kerze (das Gebiet zwischen der |
//| letzten und vorletzten Bar)                                |
//+-----+
void ClearEndOfBodyMain(const int ind)
{
    ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShadowSecond);
    ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowSecond);
};

//+-----+
//| Die Entleerung der Kerze                                    |
//+-----+
void ClearCandle(double &body_fst[],double &body_snd[],double &shadow_fst[],
                 double &shadow_snd[],const int start,const int count)
{
    //--- die Überprüfung
    if(count!=0)
    {
        //--- füllen Sie der Indikator Puffers mit dem leeren Wert aus
        ArrayFill(body_fst,start,count,INDICATOR_EMPTY_VALUE);
        ArrayFill(body_snd,start,count,INDICATOR_EMPTY_VALUE);
        ArrayFill(shadow_fst,start,count,INDICATOR_EMPTY_VALUE);
        ArrayFill(shadow_snd,start,count,INDICATOR_EMPTY_VALUE);
    }
};

//+-----+
//| Die Bildung des Hauptteiles der Kerze                      |
//+-----+
void FormMain(double &fst[],double &snd[],const double fst_value,
              const double snd_value,const int start,const int count)
{
    //--- die Überprüfung
    if(count!=0)
    {
        //--- füllen Sie die Indikator Puffers Werte aus
        ArrayFill(fst,start,count,fst_value);

```



```

        ArrayFill(snd,start,count,snd_value);
    ;}

; }

//+-----+
//| Die Bildung das Ende der Kerze |
//+-----+
void FormEnd(double &fst[],double &snd[],const double fst_value,
             const double snd_value,const int last)
{
    //--- füllen Sie die Indikator Puffers Werte aus
    ArrayFill(fst,last-1,2,fst_value);
    ArrayFill(snd,last-1,2,snd_value);
; }

//+-----+
//| Füllen Sie die i-ten Elemente des Indikator Puffers mit dem |
//| leeren Werte aus |
//+-----+
void FillIndicatorBuffers(const int i)
{
    //--- stellen Sie der leer Wert in die Zelle der Indikator Puffers ein
    ExtBearBodyFirst[i]=INDICATOR_EMPTY_VALUE;
    ExtBearBodySecond[i]=INDICATOR_EMPTY_VALUE;
    ExtBearShadowFirst[i]=INDICATOR_EMPTY_VALUE;
    ExtBearShadowSecond[i]=INDICATOR_EMPTY_VALUE;
    ExtBearBodyEndFirst[i]=INDICATOR_EMPTY_VALUE;
    ExtBearBodyEndSecond[i]=INDICATOR_EMPTY_VALUE;
    ExtBearShadowEndFirst[i]=INDICATOR_EMPTY_VALUE;
    ExtBearShadowEndSecond[i]=INDICATOR_EMPTY_VALUE;
    ExtBullBodyFirst[i]=INDICATOR_EMPTY_VALUE;
    ExtBullBodySecond[i]=INDICATOR_EMPTY_VALUE;
    ExtBullShadowFirst[i]=INDICATOR_EMPTY_VALUE;
    ExtBullShadowSecond[i]=INDICATOR_EMPTY_VALUE;
    ExtBullBodyEndFirst[i]=INDICATOR_EMPTY_VALUE;
    ExtBullBodyEndSecond[i]=INDICATOR_EMPTY_VALUE;
    ExtBullShadowEndFirst[i]=INDICATOR_EMPTY_VALUE;
    ExtBullShadowEndSecond[i]=INDICATOR_EMPTY_VALUE;
; }

```

## ArrayMinimum

Sucht im eindimensionalen numerischen Array das minimale Element.

```
int ArrayMinimum(
    const void& array[],           // das Array für die Suche
    int start=0,                  // Anfangsindex für die Suche
    int count=WHOLE_ARRAY         // Zahl der geprüften Elementen
);
```

### Parameter

`array[]`

[in] Numerisches Array, in dem die Suche durchgeführt wird.

`start=0`

[in] Anfangsindex für die Prüfung.

`count=WHOLE_ARRAY`

[in] Zahl der Elementen für Suche. Als Voreinstellung sucht im ganzen Array (count=WHOLE\_ARRAY).

### Rückgabewert

Funktion gibt Index des gefundenen Elementes unter Beachtung der Seriengröße des Arrays. Beim Misserfolg gibt die Funktion -1 zurück.

### Beispiel:

```
#property description "Der Indikator zeigt die Kerzen mit dem älteren auf die aktuelle
//--- Indikator Einstellungen
#property indicator_chart_window
#property indicator_buffers 16
#property indicator_plots 8
//---- plot 1
#property indicator_label1 "BearBody"
#property indicator_color1 clrSeaGreen,clrSeaGreen
//---- plot 2
#property indicator_label2 "BearBodyEnd"
#property indicator_color2 clrSeaGreen,clrSeaGreen
//---- plot 3
#property indicator_label3 "BearShadow"
#property indicator_color3 clrSalmon,clrSalmon
//---- plot 4
#property indicator_label4 "BearShadowEnd"
#property indicator_color4 clrSalmon,clrSalmon
//---- plot 5
#property indicator_label5 "BullBody"
#property indicator_color5 clrOlive,clrOlive
//---- plot 6
#property indicator_label6 "BullBodyEnd"
#property indicator_color6 clrOlive,clrOlive
```

```

//---- plot 7
#property indicator_label7 "BullShadow"
#property indicator_color7 clrSkyBlue,clrSkyBlue
//---- plot 8
#property indicator_label8 "BullShadowEnd"
#property indicator_color8 clrSkyBlue,clrSkyBlue
//--- vordefinierte Konstante
#define INDICATOR_EMPTY_VALUE 0.0
//--- Eingabeparameters
input ENUM_TIMEFRAMES InpPeriod=PERIOD_H4; // Der Zeitrahmen für die Berechnung
input datetime InpDateStart=D'2013.01.01 00:00'; // das Anfangsdatum der Analyse
//--- die Indikator Puffers für die Bären Kerzen
double ExtBearBodyFirst[];
double ExtBearBodySecond[];
double ExtBearBodyEndFirst[];
double ExtBearBodyEndSecond[];
double ExtBearShadowFirst[];
double ExtBearShadowSecond[];
double ExtBearShadowEndFirst[];
double ExtBearShadowEndSecond[];
//--- die Indikator Puffers für die Bullen Kerzen
double ExtBullBodyFirst[];
double ExtBullBodySecond[];
double ExtBullBodyEndFirst[];
double ExtBullBodyEndSecond[];
double ExtBullShadowFirst[];
double ExtBullShadowSecond[];
double ExtBullShadowEndFirst[];
double ExtBullShadowEndSecond[];
//--- globale Variablen
datetime ExtTimeBuff[]; // Puffer für Zeit mit der höheren Zeitrahmen
int ExtSize=0; // die Größe des Zeitpuffers
int ExtCount=0; // der Index innerhalb des Zeitpuffers
int ExtStartPos=0; // die Anfangsposition für die Berechnung des Indikators
bool ExtStartFlag=true; // Die Hilfsflagge für das Erhalten der Anfangsposition
datetime ExtCurrentTime[1]; // die letzte Zeit wenn der Bar auf dem älteren Zeitrahmen
datetime ExtLastTime; // die letzte Zeit auf dem älteren Zeitrahmen, für den die Berechnung
bool ExtBearFlag=true; // die Flagge für die Bestimmung der Ordnung der Speicherung
bool ExtBullFlag=true; // die Flagge für die Bestimmung der Ordnung der Speicherung
int ExtIndexMax=0; // der Index des maximalen Elementes in dem Array
int ExtIndexMin=0; // der Index des minimalen Elementes in dem Array
int ExtDirectionFlag=0; // die Richtung Die Bewegungen des Preises in der aktuellen Zeit
//--- der Einzug zwischen dem Eröffnungspreis und Schlusspreis der Kerze für die richtige
const double ExtEmptyBodySize=0.2*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//+-----+
//| Zeichnung der Hauptteil der Kerze |
//+-----+
void FillCandleMain(const double &open[],const double &close[],
const double &high[],const double &low[],

```

```

        const int start,const int last,const int fill_index,
        int &index_max,int &index_min)
    {
        //--- finden Sie die Indexe der maximalen und minimalen Elemente in den Arrays
        index_max=ArrayMaximum(high,ExtStartPos,last-start+1); // Maximum in Hoch
        index_min=ArrayMinimum(low,ExtStartPos,last-start+1); // Minimum in Tief
        //--- bestimmen Sie, wie viel Bars auf den aktuellen Zeitrahmen wir werden zeichnen
        int count=fill_index-start+1;
        //--- wenn der Schlusspreis am ersten Bar mehr als der Schlusspreis am letzten Bar - es
        if(open[start]>close[last])
        {
            //--- wenn früher die Kerze Bullen war, so freimachen wir die Werte den Bären Indikat
            if(ExtDirectionFlag!=-1)
                ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowSecond);
            //--- Bären Kerze
            ExtDirectionFlag=-1;
            //--- bilden Sie die Kerze
            FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShadowSecond,
                close[last],high[index_max],low[index_min],start,count,ExtBearFlag);
            //--- Funktionsausgang
            return;
        }
        //--- wenn der Schlusspreis am ersten Bar weniger als der Schlusspreis am letzten - es
        if(open[start]<close[last])
        {
            //--- wenn früher die Kerze Bären war, so freimachen wir die Werte den Bären Indikat
            if(ExtDirectionFlag!=1)
                ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShadowSecond);
            //--- Bullen Kerze
            ExtDirectionFlag=1;
            //--- bilden Sie die Kerze
            FormCandleMain(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowSecond,
                open[start],high[index_max],low[index_min],start,count,ExtBullFlag);
            //--- Funktionsausgang
            return;
        }
        //--- wenn sie in diesem Teil der Funktion, so bedeutet es, so stellt sich der Eröffnung
        //--- dem Schlusspreis am letzten Bar; wir werden solche Kerze Bären halten
        //--- wenn früher die Kerze Bullen war, so freimachen wir die Werte der Bullen Indikat
        if(ExtDirectionFlag!=-1)
            ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowSecond);
        //--- Bären Kerze
        ExtDirectionFlag=-1;
        //--- wenn Schlusspreise und Eröffnungspreise gleich sind, dann verschoben Sie für die
        if(high[index_max]!=low[index_min])
            FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShadowSecond,
                open[start]-ExtEmptyBodySize,high[index_max],low[index_min],start,count,ExtBearFlag);
        else
            FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShadowSecond,
                open[start],high[index_max],low[index_min],start,count,ExtBearFlag);
    }

```

```

        open[start],open[start]-ExtEmptyBodySize,high[index_max],
        high[index_max]-ExtEmptyBodySize,start,count,ExtBearFlag);

; }
//+-----+
//| Zeichnung das Ende der Kerze |
//+-----+
void FillCandleEnd(const double &open[],const double &close[],
                  const double &high[],const double &low[],
                  const int start,const int last,const int fill_index,
                  const int index_max,const int index_min)
{
//--- wenn es nur eine Bar gibt, dann wir nicht zeichnen
    if(last-start==0)
        return;
//--- wenn der Schlusspreis am ersten Bar mehr als der Schlusspreis am letzten Bar - es
    if(open[start]>close[last])
    {
        //--- bilden Sie das Ende der Kerze
        FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                      open[start],close[last],high[index_max],low[index_min],fill_index,
        //--- Funktionsausgang
        return;
    }
; }
//--- wenn der Schlusspreis am ersten Bar weniger als der Schlusspreis am letzten - es
    if(open[start]<close[last])
    {
        //--- bilden Sie das Ende der Kerze
        FormCandleEnd(ExtBullBodyEndFirst,ExtBullBodyEndSecond,ExtBullShadowEndFirst,Ext
                      close[last],open[start],high[index_max],low[index_min],fill_index,
        //--- Funktionsausgang
        return;
    }
; }
//--- wenn sie in diesem Teil der Funktion, so bedeutet es, so stellt sich der Eröffn
//--- dem Schlusspreis am letzten Bar; wir werden solche Kerze Bären halten
//--- bilden Sie das Ende der Kerze
    if(high[index_max]!=low[index_min])
        FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                      open[start]-ExtEmptyBodySize,high[index_max],low[index_min],fill_i
    else
        FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                      open[start]-ExtEmptyBodySize,high[index_max],high[index_max]-ExtEn
; }
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- Überprüfen Sie die Indikator Periode
    if(!CheckPeriod((int)Period(),(int)InpPeriod))

```

```

        return(INIT_PARAMETERS_INCORRECT);
//--- die Anzeige der Preisdaten auf dem Vordergrund
    ChartSetInteger(0,CHART_FOREGROUND,0,1);
//--- Bindung von Indikator Puffers
    SetIndexBuffer(0,ExtBearBodyFirst);
    SetIndexBuffer(1,ExtBearBodySecond);
    SetIndexBuffer(2,ExtBearBodyEndFirst);
    SetIndexBuffer(3,ExtBearBodyEndSecond);
    SetIndexBuffer(4,ExtBearShadowFirst);
    SetIndexBuffer(5,ExtBearShadowSecond);
    SetIndexBuffer(6,ExtBearShadowEndFirst);
    SetIndexBuffer(7,ExtBearShadowEndSecond);
    SetIndexBuffer(8,ExtBullBodyFirst);
    SetIndexBuffer(9,ExtBullBodySecond);
    SetIndexBuffer(10,ExtBullBodyEndFirst);
    SetIndexBuffer(11,ExtBullBodyEndSecond);
    SetIndexBuffer(12,ExtBullShadowFirst);
    SetIndexBuffer(13,ExtBullShadowSecond);
    SetIndexBuffer(14,ExtBullShadowEndFirst);
    SetIndexBuffer(15,ExtBullShadowEndSecond);
//--- stellen Sie einige Werte der Eigenschaften für die Konstruktion des Indikators e
    for(int i=0;i<8;i++)
    {
        PlotIndexSetInteger(i,PLOT_DRAW_TYPE,DRAW_FILLING); // der Typ der graphischen F
        PlotIndexSetInteger(i,PLOT_LINE_STYLE,STYLE_SOLID); // der Stil der Linie auf d
        PlotIndexSetInteger(i,PLOT_LINE_WIDTH,1);           // die Dicke der Linie auf c
    };
//---
    return(INIT_SUCCEEDED);
};
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{
//--- wenn es noch keine berechnete Bars gibt
    if(prev_calculated==0)
    {
        //--- bekommen Sie die Zeit wenn der Bar auf dem älteren Zeitrahmen erschienen v
        if(!GetTimeData())

```

```

        return(0);
    ;}

//--- stellen Sie die gerade Richtung der Indexierung ein
    ArraySetAsSeries(time,false);
    ArraySetAsSeries(high,false);
    ArraySetAsSeries(low,false);
    ArraySetAsSeries(open,false);
    ArraySetAsSeries(close,false);
//--- Variable zur Berechnung den Anfang des Bars
    int start=prev_calculated;
//--- wenn der Bar gebildet ist, so berechnen wir den Indikatorwert darauf
    if(start!=0 && start==rates_total)
        start--;
//--- die Zyklus von die Berechnung der Indikatorwerte
    for(int i=start;i<rates_total;i++)
    {
        //--- füllen Sie die i-ten Elemente des Indikator Puffers mit dem leeren Werten
        FillIndicatorBuffers(i);
        //--- führen Sie die Berechnung für die Bars seit dem Datum InpDateStart durch
        if(time[i]>=InpDateStart)
        {
            //--- zum ersten Mal bestimmen Sie eine Position mit der wir werden die Werte
            if(ExtStartFlag)
            {
                //--- speichern Sie der Index des Anfangsbar
                ExtStartPos=i;
                //--- bestimmen Sie das erste Datum auf dem älteren Zeitrahmen, das mehr a
                while(time[i]>=ExtTimeBuff[ExtCount])
                    if(ExtCount<ExtSize-1)
                        ExtCount++;
                //--- ändern Sie die Flaggewert, um in diesen Block nicht mehr zu geraten
                ExtStartFlag=false;
            }
            //--- prüfen Sie, ob es noch Elemente in das Array sind
            if(ExtCount<ExtSize)
            {
                //--- warten Sie, wenn der Zeitwert auf dem aktuellen Zeitrahmen der Wert
                if(time[i]>=ExtTimeBuff[ExtCount])
                {
                    //--- zeichnen Sie den Hauptteil der Kerze (ohne Zeichnung zwischen den
                    FillCandleMain(open,close,high,low,ExtStartPos,i-1,i-2,ExtIndexMax,ExtI
                    //--- zeichnen Sie das Ende der Kerze (das Gebiet zwischen der letzten
                    FillCandleEnd(open,close,high,low,ExtStartPos,i-1,i-1,ExtIndexMax,ExtI
                    //--- bewegen Sie die Anfangsposition für die Zeichnung der nächsten Ke
                    ExtStartPos=i;
                    //--- erhöhen Sie der Array Zähler
                    ExtCount++;
                }
            }
            else

```

```

        continue;
    };
else
{
    //--- der Fehlerwert zu stürzen
    ResetLastError();
    //--- bekommen Sie die letzte Datum auf dem älteren Zeitrahmen
    if(CopyTime(Symbol(), InpPeriod, 0, 1, ExtCurrentTime)==-1)
    {
        Print("Fehler beim Kopieren die Daten, Kode = ", GetLastError());
        return(0);
    };
    //--- wenn das neue Datum mehr ist, so beenden wir die Bildung der Kerze
    if(ExtCurrentTime[0]>ExtLastTime)
    {
        //--- freimachen Sie das Gebiet zwischen der letzten und vorletzten Bar
        ClearEndOfBodyMain(i-1);
        //--- zeichnen Sie dieses Gebiet mit dem Hilfs-Indikator Puffers
        FillCandleEnd(open, close, high, low, ExtStartPos, i-1, i-1, ExtIndexMax, ExtIndexMin);
        //--- bewegen Sie die Anfangsposition für die Zeichnung der nächsten Kerze
        ExtStartPos=i;
        //--- stürzen Sie die Flagge der Preisrichtung
        ExtDirectionFlag=0;
        //--- speichern Sie das neue letzte Datum
        ExtLastTime=ExtCurrentTime[0];
    };
else
{
    //--- bilden Sie die Kerze
    FillCandleMain(open, close, high, low, ExtStartPos, i, i, ExtIndexMax, ExtIndexMin);
};
};
};

//--- den Wert prev_calculated für den nächsten Anruf zurückgeben
return(rates_total);
};

//+-----+
//| Überprüfen Sie die Indikator Periode auf die Korrektheit |
//+-----+
bool CheckPeriod(int current_period, int high_period)
{
    //--- die Periode des Indikators soll mehr als der Zeitrahmen sein, auf dem er dargestellt wird
    if(current_period>=high_period)
    {
        Print("Fehler! Der Wert von der Indikator Periode soll mehr als der aktuellen Zeitrahmen sein");
        return(false);
    };
    //--- wenn die Periode des Indikators eine Woche oder einer Monat ist, so ist die Periode korrekt

```



```

    if(high_period>32768)
        return(true);
//--- bringen Sie die Werte der Perioden bis zu Minuten
    if(high_period>30)
        high_period=(high_period-16384)*60;
    if(current_period>30)
        current_period=(current_period-16384)*60;
//--- die Periode des Indikators soll Fach der Zeitraahmen sein, auf dem er dargestellt
    if(high_period%current_period!=0)
    {
        Print("Fehler! Der Wert der Periode des Indikators soll Fach der aktuellen Zeitr
        return(false);
    };
//--- die Periode des Indikators soll der Wert der Zeitraahmen übertreten, auf dem er
    if(high_period/current_period<3)
    {
        Print("Fehler! Der Wert der Periode des Indikators soll der Wert der aktuellen
        return(false);
    };
//--- die Periode des Indikators ist für den aktuellen Zeitraahmen korrekt
    return(true);
};
//+-----+
//| Bekommen Sie die Daten der Zeit auf dem älteren Zeitraahmen |
//+-----+
bool GetTimeData(void)
{
//--- der Fehlerwert zu stürzen
    ResetLastError();
//--- kopieren Sie alle Daten auf die aktuelle Zeit
    if(CopyTime(Symbol(),InpPeriod,InpDateStart,TimeCurrent(),ExtTimeBuff)==-1)
    {
        //--- der Fehlerwert zu stürzen
        int code=GetLastError();
        //--- drucken Sie der Fehlertext
        PrintFormat("Fehler beim Kopieren die Daten! %s",code==4401
            ? "Die Geschichte wird noch beladen!"
            : "Kode = "+IntegerToString(code));
        //--- geben Sie false für den nochmaligen Versuch der Beladung der Daten zurück
        return(false);
    };
//--- bekommen Sie die Größe des Arrays
    ExtSize=ArraySize(ExtTimeBuff);
//--- stellen Sie der Index des Zyklus für das Array gleich der Null ein
    ExtCount=0;
//--- stellen Sie die Position der aktuellen Kerze in diesem Zeitraahmen gleich der Null
    ExtStartPos=0;
    ExtStartFlag=true;
//--- bekommen Sie der letzten Wert der Zeit auf dem älteren Zeitraahmen

```

```

    ExtLastTime=ExtTimeBuff[ExtSize-1];
//--- die erfolgreiche Ausführung
    return(true);
; }
//+-----+
//| Die Funktion bildet den Hauptteil der Kerze. Je nach dem Wert |
//| der Flagge, die Funktion bestimmt, welche Daten sich in welche |
//| Arrays für die korrekte Abbildung einschreiben sollen.          |
//+-----+
void FormCandleMain(double &body_fst[],double &body_snd[],
                    double &shadow_fst[],double &shadow_snd[],
                    const double fst_value,const double snd_value,
                    const double fst_extremum,const double snd_extremum,
                    const int start,const int count,const bool flag)
{
//--- prüfen Sie der Wert der Flagge
    if(flag)
    {
        //--- bilden Sie der Körper der Kerze
        FormMain(body_fst,body_snd,fst_value,snd_value,start,count);
        //--- bilden Sie der Schatten der Kerze
        FormMain(shadow_fst,shadow_snd,fst_extremum,snd_extremum,start,count);
    ; }
    else
    {
        //--- bilden Sie der Körper der Kerze
        FormMain(body_fst,body_snd,snd_value,fst_value,start,count);
        //--- bilden Sie der Schatten der Kerze
        FormMain(shadow_fst,shadow_snd,snd_extremum,fst_extremum,start,count);
    ; }
; }
//+-----+
//| Die Funktion bildet das Ende der Kerze. Je nach dem Wert der |
//| Flagge, die Funktion bestimmt, welche Daten sich in welche Arrays|
//| für die korrekte Abbildung einschreiben sollen.                |
//+-----+
void FormCandleEnd(double &body_fst[],double &body_snd[],
                   double &shadow_fst[],double &shadow_snd[],
                   const double fst_value,const double snd_value,
                   const double fst_extremum,const double snd_extremum,
                   const int end,bool &flag)
{
//--- prüfen Sie der Wert der Flagge
    if(flag)
    {
        //--- bilden Sie das Ende der Körper der Kerze
        FormEnd(body_fst,body_snd,fst_value,snd_value,end);
        //--- bilden Sie das Ende der Schatten der Kerze
        FormEnd(shadow_fst,shadow_snd,fst_extremum,snd_extremum,end);
    }
}

```

```

    //--- ändern Sie der Wert der Flagge gegen das Entgegengesetzte
    flag=false;
};
else
{
    //--- bilden Sie das Ende der Körper der Kerze
    FormEnd(body_fst,body_snd,snd_value,fst_value,end);
    //--- bilden Sie das Ende der Schatten der Kerze
    FormEnd(shadow_fst,shadow_snd,snd_extremum,fst_extremum,end);
    //--- ändern Sie der Wert der Flagge gegen das Entgegengesetzte
    flag=true;
};
};

//+-----+
//| Die Entleerung das Ende der Kerze (das Gebiet zwischen der |
//| letzten und vorletzten Bar)                                |
//+-----+
void ClearEndOfBodyMain(const int ind)
{
    ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShadowSecond);
    ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowSecond);
};

//+-----+
//| Die Entleerung der Kerze                                    |
//+-----+
void ClearCandle(double &body_fst[],double &body_snd[],double &shadow_fst[],
                 double &shadow_snd[],const int start,const int count)
{
    //--- die Überprüfung
    if(count!=0)
    {
        //--- füllen Sie der Indikator Puffers mit dem leeren Wert aus
        ArrayFill(body_fst,start,count,INDICATOR_EMPTY_VALUE);
        ArrayFill(body_snd,start,count,INDICATOR_EMPTY_VALUE);
        ArrayFill(shadow_fst,start,count,INDICATOR_EMPTY_VALUE);
        ArrayFill(shadow_snd,start,count,INDICATOR_EMPTY_VALUE);
    }
};

//+-----+
//| Die Bildung des Hauptteiles der Kerze                      |
//+-----+
void FormMain(double &fst[],double &snd[],const double fst_value,
              const double snd_value,const int start,const int count)
{
    //--- die Überprüfung
    if(count!=0)
    {
        //--- füllen Sie die Indikator Puffers Werte aus
        ArrayFill(fst,start,count,fst_value);
    }
};

```

```

        ArrayFill(snd,start,count,snd_value);
    ;}

; }

//+-----+
//| Die Bildung das Ende der Kerze |
//+-----+
void FormEnd(double &fst[],double &snd[],const double fst_value,
             const double snd_value,const int last)
{
    //--- füllen Sie die Indikator Puffers Werte aus
    ArrayFill(fst,last-1,2,fst_value);
    ArrayFill(snd,last-1,2,snd_value);
; }

//+-----+
//| Füllen Sie die i-ten Elemente des Indikator Puffers mit dem |
//| leeren Werte aus |
//+-----+
void FillIndicatorBuffers(const int i)
{
    //--- stellen Sie der leer Wert in die Zelle der Indikator Puffers ein
    ExtBearBodyFirst[i]=INDICATOR_EMPTY_VALUE;
    ExtBearBodySecond[i]=INDICATOR_EMPTY_VALUE;
    ExtBearShadowFirst[i]=INDICATOR_EMPTY_VALUE;
    ExtBearShadowSecond[i]=INDICATOR_EMPTY_VALUE;
    ExtBearBodyEndFirst[i]=INDICATOR_EMPTY_VALUE;
    ExtBearBodyEndSecond[i]=INDICATOR_EMPTY_VALUE;
    ExtBearShadowEndFirst[i]=INDICATOR_EMPTY_VALUE;
    ExtBearShadowEndSecond[i]=INDICATOR_EMPTY_VALUE;
    ExtBullBodyFirst[i]=INDICATOR_EMPTY_VALUE;
    ExtBullBodySecond[i]=INDICATOR_EMPTY_VALUE;
    ExtBullShadowFirst[i]=INDICATOR_EMPTY_VALUE;
    ExtBullShadowSecond[i]=INDICATOR_EMPTY_VALUE;
    ExtBullBodyEndFirst[i]=INDICATOR_EMPTY_VALUE;
    ExtBullBodyEndSecond[i]=INDICATOR_EMPTY_VALUE;
    ExtBullShadowEndFirst[i]=INDICATOR_EMPTY_VALUE;
    ExtBullShadowEndSecond[i]=INDICATOR_EMPTY_VALUE;
; }

```

## ArrayRange

Gibt die Anzahl der Elemente in der angegebenen Arraydimension.

```
int ArrayRange(  
    const void& array[], // Array für die Prüfung  
    int rank_index // Nummer der Dimension  
);
```

### Parameter

*array[]*

[in] Das geprüfte Array.

*rank\_index*

[in] Index der Dimension.

### Rückgabewert

Anzahl der Elemente in der angegebenen Arraydimension

### Bemerkung

Da die Indizes bei Null beginnen, ist die Anzahl der Dimensionen des Arrays um eins größer als der Index der letzten Dimension.

### Beispiel:

```
void OnStart()  
{  
    //--- erstellen Sie vierdimensionalen Array  
    double array[][5][2][4];  
    //--- stellen Sie die Größe der Nullmessung  
    ArrayResize(array,10,10);  
    //--- drucken Sie die Dimensionen der Messungen  
    int temp;  
    for(int i=0;i<4;i++)  
    {  
        //--- bekommen Sie die Größe der i-ten Messung  
        temp=ArrayRange(array,i);  
        //--- drucken  
        PrintFormat("dim = %d, range = %d",i,temp);  
    }  
    //--- Ergebnis  
    // dim = 0, range = 10  
    // dim = 1, range = 5  
    // dim = 2, range = 2  
    // dim = 3, range = 4  
}
```

## ArrayResize

Stellt eine neue Größe in der ersten Dimension des Arrays ein

```
int ArrayResize(  
    void& array[],           // Array, das durch Referenz übertragen wurde  
    int new_size,           // die neue Größe des Arrays  
    int reserve_size=0      // Reservewert einer Größe (redundant)  
);
```

### Parameter

*array[]*

[out] das Array für die Veränderung der Größen

*new\_size*

[in] Die neue Größe für die erste Dimension

*reserve\_size=0*

[in] Die Größe für Zusatzreserve

### Rückgabewert

Bei erfolgreicher Durchführung gibt die Funktion die Anzahl aller Elemente zurück, die ein Array nach der Größenveränderung enthält; anderenfalls gibt -1 zurück und das Array verändert seine Größen nicht.

### Bemerkung

Die Funktion kann nur zu den [dynamischen Arrays](#) verwendet sein. Es sollte beachtet werden, dass die Größe für dynamische Arrays, die als Indikator-Puffer mit [SetIndexBuffer\(\)](#)-Funktion designiert sind, nicht verändert werden kann. Für Indikator-Puffer werden alle Operationen der Änderungen der Größe von Laufzeit-Subsystem des Terminals durchgeführt.

Общее число элементов в массиве не может превышать 2147483647.

Bei der häufiger Speicherverteilung wird empfohlen, einen dritten Parameter, der die Reserve die Anzahl der physikalischer Speicherverteilung zu reduzieren gibt. Alle nachfolgenden Aufrufe von die Funktion [ArrayResize](#) führen nicht zu physikalischer Speicherverteilung, in diesen Fällen ändert nur die Größe der ersten Dimension des Arrays in den reservierten Speicher. Es sei daran erinnert, dass der dritte Parameter nur dann verwendet wird, wenn die physische Speicherzuordnung stattfindet, zum Beispiel:

```
ArrayResize(arr,1000,1000);  
for(int i=1;i<3000;i++)  
    ArrayResize(arr,i,1000);
```

In diesem Fall werden zwei Speicherverteilungen, einmal vor dem Eingang in den Zyklus auf 2000 Elementen geschehen, dabei wird die Dimension des Arrays in 1000 und den zweiten bei i gleich 2000 eingestellt sein. Wenn der dritte Parameter weggelassen wird, passieren 2000 physikalische Speicherverteilungen und es wird die Ausführung des Programms verzögern.

### Beispiel:

```

//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//--- Zähler
    ulong start=GetTickCount();
    ulong now;
    int count=0;
//--- Array demonstriert die schnelle Version
    double arr[];
    ArrayResize(arr,100000,100000);
//--- Prüfen wir, wie schnell die Version mit Reservierung von Speicher arbeitet
    Print("--- Test Fast: ArrayResize(arr,100000,100000)");
    for(int i=1;i<=300000;i++)
    {
        //--- Geben wir eine neue Arraygröße ein mit die Reserve von 100000 Elementen!
        ArrayResize(arr,i,100000);
        //--- Wenn die runde Zahl erreicht ist, anzeige die Größe des Arrays und die Zeit
        if(ArraySize(arr)%100000==0)
        {
            now=GetTickCount();
            count++;
            PrintFormat("%d. ArraySize(arr)=%d Time=%d ms",count,ArraySize(arr),(now-start));
            start=now;
        }
    }
//--- Jetzt zeigen wir wie langsam die Version mit dem nicht-reservierten Speicher ist
    double slow[];
    ArrayResize(slow,100000,100000);
//---
    count=0;
    start=GetTickCount();
    Print("---- Test Slow: ArrayResize(slow,100000)");
//---
    for(int i=1;i<=300000;i++)
    {
        //--- Geben wir eine neue Arraygröße ohne die zusätzliche Reserve
        ArrayResize(slow,i);
        //--- Wenn die runde Zahl erreicht ist, anzeige die Größe des Arrays und die Zeit
        if(ArraySize(slow)%100000==0)
        {
            now=GetTickCount();
            count++;
            PrintFormat("%d. ArraySize(slow)=%d Time=%d ms",count,ArraySize(slow),(now-start));
            start=now;
        }
    }
}
//--- Annähernde Ausgabe des Scripts
/*
Test_ArrayResize (EURUSD,H1)    --- Test Fast: ArrayResize(arr,100000,100000)
Test_ArrayResize (EURUSD,H1)    1. ArraySize(arr)=100000 Time=0 ms
Test_ArrayResize (EURUSD,H1)    2. ArraySize(arr)=200000 Time=0 ms
Test_ArrayResize (EURUSD,H1)    3. ArraySize(arr)=300000 Time=0 ms
Test_ArrayResize (EURUSD,H1)    ---- Test Slow: ArrayResize(slow,100000)
Test_ArrayResize (EURUSD,H1)    1. ArraySize(slow)=100000 Time=0 ms
Test_ArrayResize (EURUSD,H1)    2. ArraySize(slow)=200000 Time=0 ms
Test_ArrayResize (EURUSD,H1)    3. ArraySize(slow)=300000 Time=228511 ms
*/

```

Sehen Sie auch

[ArrayInitialize](#)



## ArraySetAsSeries

Stellt Flagge AS\_SERIES für das angegebene Objekt des dynamischen Arrays ein, Indizieren der Elemente wird wie in Zeitreihen durchgeführt werden.

```
bool ArraySetAsSeries(
    const void& array[], // Array durch Referenz
    bool flag            // true bedeutet umgekehrte Reihenfolge des Indizierens
);
```

### Parameter

*array[]*

[in][out] Numerisches Array für Einstellung.

*flag*

[in] Richtung des Arrayindizierens.

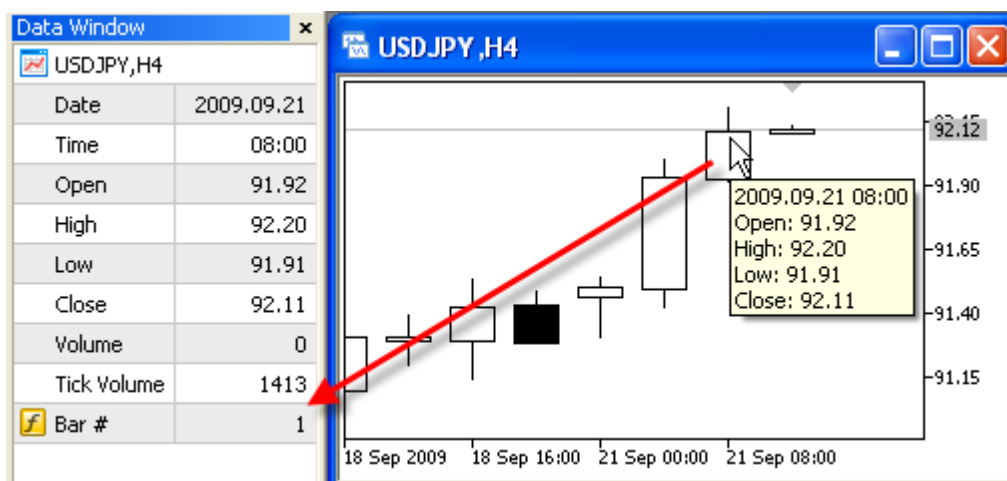
### Rückgabewert

Gibt true beim Erfolg zurück, sonst false.

### Bemerkung

Flagge AS\_SERIES kann nicht für die vieldimensionale Arrays und für die statische Arrays (d.h. Arrays, deren Größe noch während der Kompilierung in eckigen Klammern angegeben wird) eingestellt werden. Indizieren einer Zeitreihe unterscheidet sich von normalen Feld dadurch, dass Indizieren der Elemente einer Zeitreihe vom Feldende zum Feldanfang durchgeführt wird (von den neuesten Daten zu den ältesten Daten).

### Beispiel: Indikator, der Barnummer zeigt



```

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots 1
//---- plot Numeration
#property indicator_label1 "Numeration"
#property indicator_type1 DRAW_LINE
#property indicator_color1 CLR_NONE
//--- Indikator Puffer
double NumerationBuffer[];
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- Bindung von Array zum Indikator-Puffer
SetIndexBuffer(0,NumerationBuffer,INDICATOR_DATA);
//--- stellen wir Pufferindizieren wie in einer Zeitreihe ein
ArraySetAsSeries(NumerationBuffer,true);
//--- stellen wir Genauigkeit der Darstellung in DataWindow ein
IndicatorSetInteger(INDICATOR_DIGITS,0);
//--- wie der Name des Indikatorfeldes in DataWindow aussehen wird
PlotIndexSetString(0,PLOT_LABEL,"Bar #");
//---
return(INIT_SUCCEEDED);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
//--- wir werden die Eröffnungszeit der laufenden Nullbar aufbewahren
static datetime currentBarTimeOpen=0;
//--- kehren wir die Größe des Arrays time[] um - machen wir es wie in einer Zeitreihe
ArraySetAsSeries(time,true);
//--- wenn sich die Zeit der Nullbar davon unterscheidet, was wir aufbewahren
if(currentBarTimeOpen!=time[0])
{
//--- numerieren wir alle Bars von der laufenden Position in die Charttiefe
for(int i=rates_total-1;i>=0;i--) NumerationBuffer[i]=i;
currentBarTimeOpen=time[0];
}
//--- den Wert prev_calculated für den nächsten Anruf zurückgeben
return(rates_total);
}

```

Sehen Sie auch

[Zugang zu Zeitreihen](#), [ArrayGetAsSeries](#)

## ArraySize

Gibt die Anzahl von Elementen des angegebenen Arrays zurück.

```
int ArraySize(  
    const void& array[]    // das geprüfte Array  
);
```

### Parameter

*array[]*

[in] Array jedes Typs.

### Rückgabewert

Wert des Typs [int](#).

### Bemerkung

für das eindimensionale Array ist der Wert, der durch die Funktion [ArraySize](#), zurückgegeben wird dem Wert [ArrayRange](#)(array,0) gleich.

### Beispiel:

```

void OnStart()
{
//--- erstellen Sie die Arrays
    double one_dim[];
    double four_dim[][10][5][2];
//--- Größe
    int one_dim_size=25;
    int reserve=20;
    int four_dim_size=5;
//--- Hilfsvariable
    int size;
//--- trennen Sie den Speichern ohne der Reservierung
    ArrayResize(one_dim,one_dim_size);
    ArrayResize(four_dim,four_dim_size);
//--- 1. das eindimensionale Array
    Print("+=====+");
    Print("die Größe des Arrays:");
    Print("1. Das eindimensionale Array");
    size=ArraySize(one_dim);
    PrintFormat("Die Größe der Nullmessung = %d, Die Größe des Arrays = %d",one_dim_size,size);
//--- 2. das vieldimensionale Array
    Print("2. Das vieldimensionale Array");
    size=ArraySize(four_dim);
    PrintFormat("Die Größe der Nullmessung = %d, Die Größe des Arrays = %d",four_dim_size,size);
//--- die Größe der Messungen
    int d_1=ArrayRange(four_dim,1);
    int d_2=ArrayRange(four_dim,2);
    int d_3=ArrayRange(four_dim,3);
    Print("Überprüfung:");
    Print("Die Nullmessung = Die Größe des Arrays / (Die erste Messung * Die zweite Messung * Die dritte Messung)");
    PrintFormat("%d = %d / (%d * %d * %d)",size/(d_1*d_2*d_3),size,d_1,d_2,d_3);
//--- 3. das eindimensionale Array mit der Speicherreservierung
    Print("3. Das eindimensionale Array mit der Speicherreservierung");
//--- erhöhen Sie den Wert zweimal
    one_dim_size*=2;
//--- trennen Sie den Speichern mit der Reservierung
    ArrayResize(one_dim,one_dim_size,reserve);
//--- drucken Sie die Größe
    size=ArraySize(one_dim);
    PrintFormat("Die Größe mit der Reservierung = %d, Die wirkliche Größe des Arrays = %d",size,one_dim_size);
}

```

## ArraySort

Sortiert die eindimensionale numerische Arrays ansteigend von links nach rechts.

```
bool ArraySort(  
    void& array[]    // Array für Sortieren  
);
```

### Parameter

*array[]*

[in][out] Numerischer Wert für Sortieren.

### Rückgabewert

Gibt true beim Erfolg zurück, sonst false.

### Bemerkung

Array, das mit der Flagge [AS\\_SERIES](#) gekennzeichnet ist, wird absteigend sortiert.

### Beispiel:

```

#property description "Der Indikator analysiert die Daten für den letzten Monat und ze
#property description "und größten Tick Volumen aus. Für die Bestimmung solcher Kerzen
#property description "das Tick Volumen Array erzeugt. Die Kerzen, deren Volumen die e
#property description "der Prozente des Arrays bilden, gelten für die Kleinen. Die Ker
#property description "die letzte InpBigVolume der Prozente des Arrays bilden, gelten
//--- der Indikator Einstellungen
#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots 1
//--- plot
#property indicator_label1 "VolumeFactor"
#property indicator_type1 DRAW_COLOR_CANDLES
#property indicator_color1 clrDodgerBlue,clrOrange
#property indicator_style1 STYLE_SOLID
#property indicator_width1 2
//--- vordefinierte Konstante
#define INDICATOR_EMPTY_VALUE 0.0
//--- Eingabeparameters
input int InpSmallVolume=15; // Das Prozent der kleinen Volumen (<50)
input int InpBigVolume=20; // Das Prozent der größten Volumen (<50)
//--- die Anfangszeit der Analyse (wird sich verschieben)
datetime ExtStartTime;
//--- die Indikator Puffers
double ExtOpenBuff[];
double ExtHighBuff[];
double ExtLowBuff[];
double ExtCloseBuff[];
double ExtColorBuff[];
//--- die Grenzwerte der Volumen für die Abbildung der Kerzen
long ExtLeftBorder=0;
long ExtRightBorder=0;
//+-----+
//| Erhalten Sie die Grenzwerte für Tick Volumen |
//+-----+
bool GetVolumeBorders(void)
{
//--- Variablen
datetime stop_time; // die Abschlusszeit von Kopieren
long buff[]; // Der Puffer, wohin wir kopieren werden
//--- Abschlusszeit - aktuelle Zeit
stop_time=TimeCurrent();
//--- die Anfangszeit - einen Monat früher als die aktuelle
ExtStartTime=GetStartTime(stop_time);
//--- erhalten Sie die Werte der Tick Volumen
ResetLastError();
if(CopyTickVolume(Symbol(),Period(),ExtStartTime,stop_time,buff)==-1)
{
//--- Es misslang, die Daten zu bekommen, geben Sie false für den Start der Manr
PrintFormat("Es misslang, die Werte von Tick Volumen zu bekommen. Fehlercode = %
return(false);
}
//--- berechnen Sie die Größe des Arrays
int size=ArraySize(buff);
//--- sortieren Sie das Array
ArraySort(buff);
//--- bestimmen Sie die Werte der linken und der rechten Grenzen für Tick Volumen
ExtLeftBorder=buff[size*InpSmallVolume/100];
ExtRightBorder=buff[(size-1)*(100-InpBigVolume)/100];
//--- die erfolgreiche Ausführung
return(true);
}

```

```

//+-----+
//| Erhalten Sie dem Datum, einen Monat weniger als der übertragenen |
//+-----+
datetime GetStartTime(const datetime stop_time)
{
//--- konvertieren Sie die Abschlusszeit zu die Variable von die MqlDateTime-Struktur
MqlDateTime temp;
TimeToStruct(stop_time,temp);
//--- erhalten Sie dem Datum, einen Monat weniger
if(temp.mon>1)
    temp.mon-=1; // der aktuelle Monat ist nicht das erste des Jahres, d.h. die Nummer des Jahres
else
{
    temp.mon=12; // der aktuelle Monat ist das erste des Jahres, d.h. die Nummer des Jahres
    temp.year-=1; // und die Nummer des Jahres ist einer Nummer weniger
}
//--- die Nummer des tages wird nicht mehr als 28
if(temp.day>28)
    temp.day=28;
//--- geben das bekommene Datum zurück
return(StructToTime(temp));
}
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- Prüfen Sie, ob die Eingabeparameters den Bedingungen zu Zufriedenheit ausfallen
if(InpSmallVolume<0 || InpSmallVolume>=50 || InpBigVolume<0 || InpBigVolume>=50)
{
    Print("Inkorrekt Eingabeparameters");
    return(INIT_PARAMETERS_INCORRECT);
}
//--- Bindung von Array zum Indikator-Puffer
SetIndexBuffer(0,ExtOpenBuff);
SetIndexBuffer(1,ExtHighBuff);
SetIndexBuffer(2,ExtLowBuff);
SetIndexBuffer(3,ExtCloseBuff);
SetIndexBuffer(4,ExtColorBuff,INDICATOR_COLOR_INDEX);
//--- erstellen Sie die Wert, die nicht sichtbar sein werden
PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,INDICATOR_EMPTY_VALUE);
//--- stellen Sie die Beschriftungen für die Indikator Puffers
PlotIndexSetString(0,PLOT_LABEL,"Open;High;Low;Close");
//---
return(INIT_SUCCEEDED);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
//--- prüfen Sie, ob es noch die unbearbeiteten Bars sind

```

```

    if(prev_calculated<rates_total)
    {
        //--- bekommen Sie die neue Werte der linken und rechten Grenzen für Volumen
        if(!GetVolumeBorders())
            return(0);
    }
    //---Variable zur Berechnung den Anfang des Bars
    int start=prev_calculated;
    //--- wenn die Indikatorwerte auf vorhergehend Tick schon berechnet waren, so ist auf c
    if(start>0)
        start--;
    //--- stellen Sie die direkte Richtung der Indizierung in Zeitspanne
    ArraySetAsSeries(time,false);
    ArraySetAsSeries(open,false);
    ArraySetAsSeries(high,false);
    ArraySetAsSeries(low,false);
    ArraySetAsSeries(close,false);
    ArraySetAsSeries(tick_volume,false);
    //--- der Zyklus für Berechnung der Indikatorwerte
    for(int i=start;i<rates_total;i++)
    {
        //--- zeichnen Sie die Kerzen seit dem Anfangsdatum
        if(ExtStartTime<=time[i])
        {
            //--- wenn der Wert ist nicht weniger als die rechte Grenze, dann zeichnen S
            if(tick_volume[i]>=ExtRightBorder)
            {
                //--- bekommen Sie die Daten um die Kerze zu zeichnen
                ExtOpenBuff[i]=open[i];
                ExtHighBuff[i]=high[i];
                ExtLowBuff[i]=low[i];
                ExtCloseBuff[i]=close[i];
                //--- die Farbe DodgerBlue
                ExtColorBuff[i]=0;
                //--- fahren Sie die Zyklus fort weiter
                continue;
            }
            //--- wenn dir Wert ist nicht mehr als die linke Grenze, dann zeichnen Sie d
            if(tick_volume[i]<=ExtLeftBorder)
            {
                //--- bekommen Sie die Daten um die Kerze zu zeichnen
                ExtOpenBuff[i]=open[i];
                ExtHighBuff[i]=high[i];
                ExtLowBuff[i]=low[i];
                ExtCloseBuff[i]=close[i];
                //--- die Farbe Orange
                ExtColorBuff[i]=1;
                //--- fahren Sie die Zyklus fort weiter
                continue;
            }
        }
        //--- für die Bars, die nicht in die Berechnung gerieten, stellen Sie der leere
        ExtOpenBuff[i]=INDICATOR_EMPTY_VALUE;
        ExtHighBuff[i]=INDICATOR_EMPTY_VALUE;
        ExtLowBuff[i]=INDICATOR_EMPTY_VALUE;
        ExtCloseBuff[i]=INDICATOR_EMPTY_VALUE;
    }
    //--- den Wert prev_calculated für den nächsten Anruf zurückgeben
    return(rates_total);
}

```





## Datenverarbeitung

Gruppe von Funktionen, die die Angaben des bestimmten Formats in die Angaben des anderen Formats umwandelt.

Die Funktion [NormalizeDoubles\(\)](#) muss besonders betont werden, denn sie gewährleistet die erforderliche Genauigkeit der Preisdarstellung. In Handelsoperationen können nicht normalisierte Preise, deren Genauigkeit um wenigstens 1 Zeichen die vom Handelsserver angeforderte Genauigkeit überschreitet, nicht verwendet werden.

| Funktion                           | Massnahme                                                                                                                     |
|------------------------------------|-------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">CharToString</a>       | Umwandlung des Symbolcodes in die Einsymbolzeile                                                                              |
| <a href="#">DoubleToString</a>     | Umwandlung des numerischen Wertes in die Textzeile mit der angegebenen Genauigkeit                                            |
| <a href="#">EnumToString</a>       | Umwandlung der Aufzählungswert beliebigen Typs in ein String                                                                  |
| <a href="#">NormalizeDouble</a>    | Runden der Zahl mit dem Fließpunkt bis der angegebenen Genauigkeit                                                            |
| <a href="#">StringToDouble</a>     | Umwandlung der Zeile mit der Symboldarstellung der Zahl in die Zahl des Typs double                                           |
| <a href="#">StringToInteger</a>    | Umwandlung der Zeile mit der Symboldarstellung der Zahl in die Zahl des Typs int                                              |
| <a href="#">StringToTime</a>       | Umwandlung der Zeile mit Zeit und/oder Datum im Format "yyyy.mm.dd [hh:mi]", in eine Zahl des Typs datetime                   |
| <a href="#">TimeToString</a>       | Umwandlung des Wertes mit der Zeit in Sekunden, die seit 01.01.1970 vergangen hat, in die Zeile mit Format "yyyy.mm.dd hh:mi" |
| <a href="#">IntegerToString</a>    | Umwandlung des ganzzahligen Wertes in die Zeile der vorgegebenen Länge                                                        |
| <a href="#">ShortToString</a>      | Umwandlung des Symbolcodes (unicode) in die Einsymbolzeile                                                                    |
| <a href="#">ShortArrayToString</a> | Kopiert Teil des Feldes in die Zeile                                                                                          |
| <a href="#">StringToShortArray</a> | Kopiert die Zeile in den gewählten Platz des Feldes des Typs ushort                                                           |
| <a href="#">CharArrayToString</a>  | Wandelt Symbolcode (ansi) in Einsymbolzeile um                                                                                |
| <a href="#">StringToCharArray</a>  | Kopiert die Zeile, die aus Unicode in ANSI umgewandelt wurde, in den gewählten Platz des Typs uchar                           |

|                                      |                                                                                         |
|--------------------------------------|-----------------------------------------------------------------------------------------|
| <a href="#"><u>ColorToARGB</u></a>   | Konvertiert den Typ color in den Typ uint, um Farbdarstellung ARGB zu erhalten.         |
| <a href="#"><u>ColorToString</u></a> | Wandelt den Farbenwert in die Zeile der Art "R,G,B" um                                  |
| <a href="#"><u>StringToColor</u></a> | Wandelt Zeile des Typs "R,G,B" oder Zeile mit Farbennamen in den Wert des Typs color um |
| <a href="#"><u>StringFormat</u></a>  | Wandelt Zahl in die Zeile entsprechend dem vorgegebenen Format um                       |

Sehen Sie auch

[Kodeseite Verwenden](#)

## CharToString

Umwandlung des Symbolkodes in Einsymbolzeile.

```
string CharToString(  
    uchar char_code    // numerischer Symbolkode  
);
```

### Parameter

*char\_code*

[in] Kode des Symbols ANSI.

### Rückgabewert

Zeile mit dem Symbol ANSI.

## CharArrayToString

Kopiert wandelt Teil des Feldes des Typs uchar in die Rückgabezeile um.

```
string CharArrayToString(  
    uchar    array[],           // Feld  
    int      start=0,          // Anfangsposition im Feld  
    int      count=-1          // Anzahl der Symbole  
    uint     codepage=CP_ACP    // Kodeseite  
);
```

### Parameter

*array[]*

[in] Feld des Typs uchar.

*start=0*

[in] Position, aus der Kopieren anfaengt. Default-Wert ist 0.

*count=-1*

[in] Zahl der Feldelementen für Kopieren. Bestimmt die Laenge der ergebenden Zeile. Default-Wert ist -1, was bedeutet Kopieren bis zum Feldende, oder bis zum Treffen der terminalen 0.

*codepage=CP\_ACP*

[in] Wert der Kodeseite. Für die verbreitesten [Kodeseiten](#) werden entsprechende Konstanten vorausgesehen.

### Rückgabewert

Zeile.

### Sehen Sie auch

[Kodeseite Verwenden](#)

## ColorToARGB

Konvertiert den Typ [color](#) in den Typ [uint](#) um Farbdarstellung ARGB zu erhalten. Format ARGB wird für Erstellung einer [graphischen Ressource](#), [Textausgabe](#) und in der CCanvas-Klasse der Standardbibliothek.

```
uint ColorToARGB(
    color clr,           // Die Farbe im Format color, die Sie konvertieren wollen
    uchar alpha=255     // Alpha-Kanal, der Farbtransparenz verwaltet
);
```

### Optionen

*clr*

[in] Farbwert in der Variable vom Typ color.

*alpha*

[in] Wert des Alpha-Kanals, um die Farbe im Format [ARGB](#) zu erhalten. Es wird durch den Wert von 0 (Farbe des anlegenden Pixels ändert nicht die Farbe des zugrunde liegenden Pixels) bis 255 (Farbe des anlegenden Pixels voll bedeckt die Farbe des zugrunde liegenden Pixels) angegeben. Farbtransparenz in Prozent wird als  $(1 - \alpha / 255) * 100\%$  berechnet, d.h. je weniger der Alpha-Wert ist, desto mehr transparent die Farbe ist.

### Rückgabewert

Farbdarstellung im Format ARGB, wo in vier Bytes vom Typ uint Werte Alfa, Red, Green, Blue (Alpha, rot, grün, blau) vorgeschrieben sind.

### Hinweis

Das Grundformat für Beschreibung der Pixelfarbe auf dem Bildschirm in Computergrafik ist RGB, wo die Farbkomponenten nach den Namen der Primärfarben Rot (Red), Grün (Green) und Blau (Blue) gesetzt werden. Jede Komponente wird durch ein einzelnes Byte, das die Sättigung der Farbe im Bereich von 0 bis 255 (0x00 bis 0xFF hexadezimal) spezifiziert, beschrieben. Da Weiß enthält alle Farben, wird sie als 0xFFFFFF beschrieben, das heißt, jede der drei Komponenten ist im maximalen Wert 0xFF repräsentiert.

Aber in einer Reihe von Aufgaben erforderlich ist, Farbtransparenz anzugeben, um zu beschreiben, wie das Bild aussieht, wenn die Farbe mit einem gewissen Transparenz auf das Bild gelegt ist. In solchen Fällen wird das Konzept der alpha-Kanal (Alpha) benutzt, der als zusätzliche Komponente zum Format RGB eingeführt wird. Schema vom ARGB Format ist unten gezeigt.

| 8     |    |    |    |    |    |    |    | 8   |    |    |    |    |    |    |    | 8     |    |    |    |    |    |   |   | 8    |   |   |   |   |   |   |   |
|-------|----|----|----|----|----|----|----|-----|----|----|----|----|----|----|----|-------|----|----|----|----|----|---|---|------|---|---|---|---|---|---|---|
| Alpha |    |    |    |    |    |    |    | Red |    |    |    |    |    |    |    | Green |    |    |    |    |    |   |   | Blue |   |   |   |   |   |   |   |
| 31    | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23  | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15    | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7    | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

ARGB-Werte werden typischerweise in hexadezimal Format angegeben, bei dem jedes Paar von Ziffern den Wert der Kanälen Alpha, Red, Green und Blue repräsentiert. Zum Beispiel ARGB-Farbe 80FFFF00 bedeutet gelb mit Opazität von 50,2%. Am Anfang gibt es 0x80, was 50,2% Alpha-Wert angibt, da es 50,2% vom Wert 0xFF ist; weitere bedeutet erste Paar FF den maximalen Wert der roten Komponente; das nächste Paar FF gibt der gleiche Effekt der grünen Komponente an; und das letzte Paar 00 ist der Mindestwert von Blauanteil (kein blau). Die Zugabe von grün und rot gibt gelb.

Wenn der Alpha-Kanal nicht verwendet wird, kann die Aufnahme zu 6 Ziffern RRGGBB reduziert werden, weshalb die Werte des Alpha-Kanals in den oberen Bits der Integer-Typ uint gespeichert werden.

Je nach dem Kontext können die Hexadezimalzahlen mit dem Präfix '0x' oder '#', beispielsweise 80FFFF00 oder 0x80FFFF00 oder #80FFFF00 aufgezeichnet werden.

#### Beispiel:

```
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    //-- Transparenz angeben
    uchar alfa=0x55; // Wert x55 bedeutet 55/255=21.6 % von Transparenz
    //-- Konvertierung in ARGB für clrBlue
    PrintFormat("0x%.8X - clrBlue",clrBlue);
    PrintFormat("0x%.8X - clrBlue ARGB with alfa=0x55 (transparency 21.6%)",ColorToARGB);
    //-- Konvertierung in ARGB für clrGreen
    PrintFormat("0x%.8X - clrGreen",clrGreen);
    PrintFormat("0x%.8X - clrGreen ARGB with alfa=0x55 (transparency 21.6%)",ColorToARGB);
    //-- Konvertierung in ARGB für clrRed
    PrintFormat("0x%.8X - clrRed",clrRed);
    PrintFormat("0x%.8X - clrRed ARGB with alfa=0x55 (transparency 21.6%)",ColorToARGB);
}
```

#### Sehen Sie auch

[Ressourcen](#), [ResourceCreate\(\)](#), [TextOut\(\)](#), [Typ color](#), [Typen char, short, int und long](#)

## ColorToString

Wandelt Farbenwert in die Zeile der Art "R,G,B" um.

```
string ColorToString(  
    color  color_value,      // Farbenwert  
    bool   color_name       // Farbennamen zeigen oder nicht  
);
```

### Parameter

*color\_value*

[in] Farbenwert in der Variable des Typs color.

*color\_name*

[in] Zeichen der Notwendigkeit, Farbennamen rückzugeben, falls Farbenwert mit dem Wert einer der vorbestimmten [Farbenkonstanten](#) zusammenfaellt.

### Rückgabewert

Zeildarstellung der Farbe als "R,G,B", wo R, G und B Dezimalkonstanten sind, die in die Zeile umgewandelt sind und den wert im Bereich von 0 bis 255 haben. Wenn der Parameter color\_name=true eingestellt wird, wird es versucht, Farbenwert zum Farbennamen zu reduzieren.

### Beispiel:

```
string clr=ColorToString(C'0,255,0'); // grüne Farbe  
Print(clr);  
  
clr=ColorToString(C'0,255,0',true);   // Farbenkonstante erhalten  
Print(clr);
```



## DoubleToString

Umwandeln des numerischen Wertes in die Textzeile.

```
string DoubleToString(  
    double value,      // Zahl  
    int    digits=8    // Anzahl der Dezimalzeichen  
);
```

### Parameter

*value*

[in] Wert mit dem Fließpunkt.

*digits*

[in] Format der Genauigkeit. Wenn sich der Wert *digits* im Bereich von 0 bis 16 befindet, bekommt man die Zeilendarstellung der Zahl mit der angegebenen Zahl der Dezimalzeichen. Wenn sich der Wert *digits* im Bereich von -1 bis -16 befindet, bekommt man Zeilendarstellung der Zahl im wissenschaftlichen Format mit der angegebenen Anzahl der Dezimalzeichen nach dem Komma. In allen anderen Fällen wird der Zeilenwert der Zahl 8 Dezimalzeichen nach dem Komma enthalten.

### Rückgabewert

Zeile mit der Symboldarstellung der Zahl mit der angegebenen Genauigkeit.

### Beispiel:

```
Print("DoubleToString(120.0 + M_PI) : ",DoubleToString(120.0+M_PI));  
Print("DoubleToString(120.0 + M_PI,16) : ",DoubleToString(120.0+M_PI,16));  
Print("DoubleToString(120.0 + M_PI,-16) : ",DoubleToString(120.0+M_PI,-16));  
Print("DoubleToString(120.0 + M_PI,-1) : ",DoubleToString(120.0+M_PI,-1));  
Print("DoubleToString(120.0 + M_PI,-20) : ",DoubleToString(120.0+M_PI,-20));
```

### Sehen Sie auch

[NormalizeDouble](#), [StringToDouble](#)

## EnumToString

Umwandlung der Aufzählungswert beliebigen Typs in die Textdarstellung.

```
string EnumToString(  
    any_enum value    // Aufzählungswert beliebigen Typs  
);
```

### Parameter

*value*

[in] Aufzählungswert beliebigen Typs.

### Rückgabewert

String mit der Textdarstellung der Wert. Um Information über Fehler zu bekommen, rufen Sie die Funktion [GetLastError\(\)](#).

### Bemerkung

Die Funktion kann die folgende folgende Fehlerwerte in die [\\_LastError](#) Funktion einstellen:

- ERR\_INTERNAL\_ERROR - Fehler der Ausführungsumgebung
- ERR\_NOT\_ENOUGH\_MEMORY - nicht genügend Speicher für die Operation
- ERR\_INVALID\_PARAMETER - kann nicht den Namen des Enumerationswert zulassen

Beispiel:

```

enum interval // enum benannten Konstanten
{
    month=1,      // Abstand von einem Monat
    two_months,   // Zwei Monate
    quarter,      // drei Monaten - Quartal
    halfyear=6,   // Halbjahr
    year=12,      // Jahr - 12 Monate
};
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    //--- das Zeitintervall gleich zu einem Monat
    interval period=month;
    Print(EnumToString(period)+"="+IntegerToString(period));

    //--- das Zeitintervall gleich zu einem Quartal (Drei Monate)
    period=quarter;
    Print(EnumToString(period)+"="+IntegerToString(period));

    //--- das Zeitintervall gleich zu einem Jahr (12 Monate)
    period=year;
    Print(EnumToString(period)+"="+IntegerToString(period));

    //--- prüfen, wie die Auftragsart wird gezeigt
    ENUM_ORDER_TYPE type=ORDER_TYPE_BUY;
    Print(EnumToString(type)+"="+IntegerToString(type));

    //--- prüfen, wie eine falsche Wert wird gezeigt
    type=WRONG_VALUE;
    Print(EnumToString(type)+"="+IntegerToString(type));

    // Ergebnis:
    // month=1
    // quarter=3
    // year=12
    // ORDER_TYPE_BUY=0
    // ENUM_ORDER_TYPE::-1=-1
}

```

Sehen Sie auch

[Aufzählungen](#), [Input Variablen](#)

## IntegerToString

Wandelt den Wert des ganzzahligen Types in die Zeile der angegebenen Laenge um und gibt die erhaltene Zeile zurück.

```
string IntegerToString(  
    long    number,           // Zahl  
    int     str_len=0,        // Laenge der ergebenden Zeile  
    ushort  fill_symbol=' '   // Fueller  
);
```

### Parameter

*number*

[in] Zahl für die Umwandlung.

*str\_len=0*

[in] Zeilenlaenge. Wenn die Laenge der ergebenden Zeile mehr als die angegebene ist, wird die Zeile nicht reduziert. Wenn die Laenge der ergebenden Zeile weniger ist, werden Fueller-Symbole links zugesetzt.

*fill\_symbol=' '*

[in] Fueller-Symbol. Default-Wert - Space.

### Rückgabewert

Zeile.

## ShortToString

Wandelt Symbolcode (unicode) in Einsymbolzeile um und gibt die ergebende Zeile zurück.

```
string ShortToString(  
    ushort symbol_code    // Symbol  
);
```

### Parameter

*symbol\_code*

[in] Symbolcode. Statt des Symbolkodes kann eine Literalzeile mit dem Symbol verwendet werden oder eine literale Zeile mit dem 2-Byte hexadezimalen Kodex verwendet werden, der dem Symbol aus der Tabelle Unicode entspricht.

### Rückgabewert

Zeile.

## ShortArrayToString

Kopiert Teil des Feldes in die Rückgabezeile.

```
string ShortArrayToString(  
    ushort array[],      // Feld  
    int start=0,         // Anfangsposition im Feld  
    int count=-1         // Anzahl der Symbole  
);
```

### Parameter

*array[]*

[in] Feld des Typs ushort (Analog des Typs wchar\_t).

*start=0*

[in] Position, mit der Kopieren beginnt. Default-Wert ist 0.

*count=-1*

[in] Anzahl der Feldelemente für Kopieren. Bestimmt die Länge der ergebenden Zeile. Default-Wert ist -1, was bedeutet Kopieren bis zum Feldende, oder bis zur terminalen 0.

### Rückgabewert

Zeile.

## TimeToString

Umwandlung des Wertes, der Zeit in Sekunden enthält, die seit 01.01.1970 vergangen ist, in die Zeile dem Format "yyyy.mm.dd hh:mi".

```
string TimeToString(  
    datetime value,                // Zahl  
    int mode=TIME_DATE|TIME_MINUTES // Ausgabeformat  
);
```

### Parameter

*value*

[in] Zeit in Sekunden seit 00:00 1. Januar 1970.

*mode=TIME\_DATE|TIME\_MINUTES*

[in] Zusätzliches Verfahren der Datenausgabe. Kann eine oder eine kombinierte Flagge sein:

TIME\_DATE bekommt das Ergebnis in der Form " yyyy.mm.dd " ,

TIME\_MINUTES bekommt das Ergebnis in der Form " hh:mm " ,

TIME\_SECONDS bekommt das Ergebnis in der Form " hh:mm:ss ".

### Rückgabewert

Zeile.

## NormalizeDouble

Runden der Zahl mit dem Fließpunkt bis zur angegebenen Genauigkeit.

```
double NormalizeDouble(
    double value,      // normalisierte Zahl
    int    digits      // Anzahl der Dezimalzeichen
);
```

### Parameter

*value*

[in] Wert mit dem Fließpunkt.

*digits*

[in] Format der Genauigkeit, Anzahl der Dezimalzeichen (0-8).

### Rückgabewert

Wert des Typs double mit der vorgegebenen Genauigkeit.

### Bemerkung

Berechnete Werte von StopLoss, TakeProfit, und Werte des Eröffnungspreises der Warteordern, müssen normalisiert werden mit der Genauigkeit, deren Wert durch die Funktion [Digits\(\)](#) erhalten werden kann.

Bitte beachten Sie, dass bei der Ausgabe einer normalisierten Zahl ins Journal mit Funktion Print(), kann sie eine größere Anzahl von Nachkommastellen als Sie erwarten, enthalten. Zum Beispiel für:

```
double a=76.671;           // Normalisierte Zahl mit 3 Nachkommastellen
Print("Print(76.671)=",a); // Wir geben es, wie es ist, aus
Print("DoubleToString(a,8)=",DoubleToString(a,8)); // Wir geben es mit einer vorgeg
```

erhalten sie das folgende im Terminal:

```
DoubleToString(a,8)=76.67100000
```

```
Print(76.671)=76.671000000000001
```

### Beispiel:

```
double pi=M_PI;
Print("pi = ",DoubleToString(pi,16));

double pi_3=NormalizeDouble(M_PI,3);
Print("NormalizeDouble(pi,3) = ",DoubleToString(pi_3,16))
;
double pi_8=NormalizeDouble(M_PI,8);
Print("NormalizeDouble(pi,8) = ",DoubleToString(pi_8,16));

double pi_0=NormalizeDouble(M_PI,0);
Print("NormalizeDouble(pi,0) = ",DoubleToString(pi_0,16));
/*
```



```
Ergebnis:  
pi= 3.1415926535897931  
NormalizeDouble(pi,3)= 3.1419999999999999  
NormalizeDouble(pi,8)= 3.1415926499999998  
NormalizeDouble(pi,0)= 3.0000000000000000  
*/
```

**Sehen Sie auch**

[DoubleToString](#), [Realtypen \(double, float\)](#), [Typenreduzierung](#),

## StringToCharArray

Kopiert die Zeile, die aus unicode in ansi umgewandelt wurde, in den gewählten Platz des Feldes des Typs symbolenweise. Funktion gibt die Anzahl der kopierten Elemente zurück.

```
int StringToCharArray(  
    string  text_string,           // Zeile-Quelle  
    uchar&  array[],              // Feld  
    int     start=0,              // Anfangsposition im Feld  
    int     count=-1              // Anzahl der Symbole  
    uint    codepage=CP_ACP       // Kodeseite  
);
```

### Parameter

*text\_string*

[in] Zeile für Kopieren.

*array[]*

[out] Feld des Typs uchar.

*start=0*

[in] Position, mit der Kopieren beginnt. Default-Wert ist 0.

*count=-1*

[in] Anzahl der Feldelemente für Kopieren. Bestimmt die Länge der ergebenden Zeile. Default-Wert ist -1, was bedeutet Kopieren bis zum Feldende oder bis zur terminalen 0. Die terminale 0 wird auch in Feld-Rezipient kopiert werden, dabei wird gegebenenfalls die Größe des dynamischen Feldes bis zur Zeilengröße vergrößert werden. Wenn die Größe des dynamischen Feldes höher als die Zeilenlänge ist, wird die Feldgröße nicht reduziert werden.

*codepage=CP\_ACP*

[in] Wert der Kodeseite. Für die verbreitesten [Kodeseiten](#) werden entsprechende Konstanten vorgesehen.

### Rückgabewert

Anzahl der kopierten Elemente.

### Sehen Sie auch

[Kodeseite Verwenden](#)

## StringToColor

Wandelt die Zeile des Typs "R,G,B" oder die Zeile mit dem Farbennamen, in den Wert des Typs color um.

```
color StringToColor(  
    string color_string    // Zeilendarstellung der Farbe  
);
```

### Parameter

*color\_string*

[in] Zeilendarstellung der Farbe des Typs "R,G,B" oder Name einer der vorbestimmten [Web-Farben](#).

### Rückgabewert

Farbenwert.

### Beispiel:

```
color str_color=StringToColor("0,127,0");  
Print(str_color);  
Print((string)str_color);  
//--- verändern wir die Farbe ein bisschen  
str_color=StringToColor("0,128,0");  
Print(str_color);  
Print((string)str_color);
```

## StringToDouble

Umwandlung der Zeile, die Symboldarstellung der Zahl enthält, in die Zahl des Typs double.

```
double StringToDouble(  
    string value    // Zeile  
);
```

### Parameter

*value*

[in] Zeile mit der Symbolendarstellung der Zahl.

### Rückgabewert

Wert des Typs double.

## StringToInteger

Umwandlung der Zeile, die Symboldarstellung der Zahl enthält, darunter des Typs int (ganzzahlig).

```
long StringToInteger(  
    string value    // Zeile  
);
```

### Parameter

*value*

[in]die Zahl enthaltende Zeile

### Rückgabewert

Wert des Typs long.

## StringToShortArray

Kopiert die Zeile symbolenweise in den angegebenen Platz des Feldes des Typs `ushort`. Funktion gibt die Anzahl der kopierten Elemente zurück.

```
int StringToShortArray(  
    string text_string,    // Zeile-Quelle  
    ushort& array[],       // Feld  
    int start=0,          // Anfangsposition im Feld  
    int count=-1           // Anzahl der Symbole  
);
```

### Parameter

*text\_string*

[in] Zeile für Kopieren.

*array[]*

[out] Feld des Typs ushort (Analog des Typs `wchar_t`).

*start=0*

[in] Position, mit der Kopieren beginnt. Default-Wert ist 0.

*count=-1*

[in] Anzahl der Feldelemente für Kopieren. Bestimmt die Länge der ergebenden Zeile. Default-Wert ist -1, was bedeutet Kopieren bis zum Feldende oder bis zur terminalen 0. Die terminale 0 wird auch in Feld-Rezipient kopiert werden, dabei kann die Größe des dynamischen Feldes bis zur Länge der Zeile steigen. Wenn die Größe des dynamischen Feldes mehr als die Zeilenlänge ist, wird die Feldgröße nicht reduziert werden.

### Rückgabewert

Anzahl der kopierten Elemente.

## StringToTime

Umwandlung der Zeile, die Zeit und/oder Datum im Format "yyyy.mm.dd [hh:mi]" enthält, in die Zahl des Typs `datetime`.

```
datetime StringToTime (  
    string value      // Datumzeile  
);
```

### Parameter

*value*

[in] Zeile im Format " yyyy.mm.dd hh:mi ".

### Rückgabewert

Wert des Typs [datetime](#), die die Anzahl der Sekunden enthält, die seit 01.01.1970 vergangen sind.

## StringFormat

Formatiert die erhaltenen Daten und gibt Zeile zurück.

```
string StringFormat(  
    string format,      // Zeile mit Formatbeschreibung  
    ...                // Parameter  
);
```

### Parameter

*format*

[in] Zeile, die Formatierungsverfahren enthält. Regel der Formatierung sind dieselben wie die für die Funktion [PrintFormat](#)

...

[in] Parameter, die durch Komma getrennt sind

Zeile.

### Sehen Sie auch

[PrintFormat](#), [DoubleToString](#), [ColorToString](#), [TimeToString](#)



## Mathematische Funktionen

Satz der mathematischen und trigonometrischen Funktionen.

| Funktion                                 | Massnahme                                                            |
|------------------------------------------|----------------------------------------------------------------------|
| <a href="#"><u>MathAbs</u></a>           | Gibt absoluten Wert (Modulus) der angegebenen Zahl zurück            |
| <a href="#"><u>MathArccos</u></a>        | Gibt arccos x in Radianen zurück                                     |
| <a href="#"><u>MathArcsin</u></a>        | gibt arcsin x in Radianen zurück                                     |
| <a href="#"><u>MathArctan</u></a>        | Gibt arctg x in Radianen zurück                                      |
| <a href="#"><u>MathCeil</u></a>          | Gibt den ganzzahligen Wert, der der naechste von oben ist, zurück    |
| <a href="#"><u>MathCos</u></a>           | Gibt Kosinus zurück                                                  |
| <a href="#"><u>MathExp</u></a>           | Gibt Exponente der Zahl zurück                                       |
| <a href="#"><u>MathFloor</u></a>         | Gibt den ganzzahligen Wert, der der naechste von unten ist, zurück   |
| <a href="#"><u>MathLog</u></a>           | Gibt Napierschen Logarithmus zurück                                  |
| <a href="#"><u>MathLog10</u></a>         | Gibt Dezimallogarithmus zurück                                       |
| <a href="#"><u>MathMax</u></a>           | Gibt maximalen Wert der zwei numerischen Werte                       |
| <a href="#"><u>MathMin</u></a>           | Gibt minimalen Wert der zwei numerischen Werte                       |
| <a href="#"><u>MathMod</u></a>           | Gibt reellen überbleibsel der Division der zwei Zahlen               |
| <a href="#"><u>MathPow</u></a>           | Erhebt die Basis in die angegebene Potenz                            |
| <a href="#"><u>MathRand</u></a>          | Gibt die pseudozufällige Zahl im Bereich von 0 bis 32767 zurück      |
| <a href="#"><u>MathRound</u></a>         | Rundet die Zahl bis die naechste Ganzzahl                            |
| <a href="#"><u>MathSin</u></a>           | Gibt sin der Zahl zurück                                             |
| <a href="#"><u>MathSqrt</u></a>          | Gibt Quadratwurzel zurück                                            |
| <a href="#"><u>MathSrand</u></a>         | Stellt Anfangszustand für Geberieren der pseudozufälligen Realzahlen |
| <a href="#"><u>MathTan</u></a>           | Gibt tg der Zahl zurück                                              |
| <a href="#"><u>MathIsValidNumber</u></a> | Prueft Richtigkeit der Realzahl                                      |

## MathAbs

Gibt absoluten (Mod) Wert (Modulwert) der ihr übertragenen Zahl.

```
double MathAbs(  
    double value    // Zahl  
);
```

### Parameter

*value*

[in] Zahlenwert.

### Rückgabewert

Wert des Typs double, grösser als oder gleich 0.

### Bemerkung

Statt der Funktion MathAbs() kann die Funktion [fabs\(\)](#) verwendet werden.

## MathArccos

Gibt den Wert von  $\arccos x$  im Bereich 0 zu  $\pi$  in Radianen.

```
double MathArccos(  
    double val    // -1<val<1  
) ;
```

### Parameter

*val*

[in] Wert val zwischen -1 und 1, dessen arccos berechnet werden muss.

### Rückgabewert

Arccos der Zahl in Radianen. Wenn val weniger als -1 oder mehr als 1 ist, gibt die Funktion NaN (unbestimmten Wert) zurück.

### Bemerkung

Statt der Funktion MathArccos() kann die Funktion [acos\(\)](#) verwendet werden.

### Sehen Sie auch

[Realtypen \(double, float\)](#)

## MathArcsin

Gibt  $\arcsin x$  im Bereich von  $-\pi/2$  bis  $\pi/2$  Radiane.

```
double MathArcsin(  
    double val      // -1<value<1  
);
```

### Parameter

*val*

[in] Wert *val* zwischen -1 und 1, dessen arcsin berechnet werden muss.

### Rückgabewert

Arcsin *val* in Radianen im Bereich von  $-\pi/2$  bis  $\pi/2$  Radiane. Wenn *val* weniger als -1 oder mehr als 1, gibt die Funktion NaN (unbestimmten Wert) zurück.

### Bemerkung

Statt der Funktion `MathArcsin()` kann die Funktion `asin()` verwendet werden.

### Sehen Sie auch

[RealTypen \(double, float\)](#)

## MathArctan

Gibt arctangens x zurück. Wenn x gleich 0 ist, gibt die Funktion 0 zurück.

```
double MathArctan(  
    double value    // Tangens  
) ;
```

### Parameter

*value*

[in] Zahl, die Tangens vertritt.

### Rückgabewert

MathArctan gibt Wert im Bereich von  $-\pi/2$  bis  $\pi/2$  Radiane.

### Bemerkung

Statt der Funktion `MathArctan()` kann die Funktion `atan()` verwendet werden).

## MathCeil

Gibt den ganzzahligen Wert, der der naechste von oben ist, zurück.

```
double MathCeil(  
    double val    // Zahl  
);
```

### Parameter

*val*

[in] Numerischer Wert.

### Rückgabewert

Numerischer Wert, der die kleinste Ganzzahl vertritt, die groesser als oder gleich *val* ist.

### Bemerkung

Statt der Funktion MathCeil() kann die Funktion [ceil\(\)](#) verwendet werden.

## MathCos

Funktion gibt Kosinus des Winkels zurück.

```
double MathCos(  
    double value    // Zahl  
);
```

### Parameter

*value*

[in] Winkel in Radianen.

### Rückgabewert

Wert des Typs double im Bereich von -1 bis 1.

### Bemerkung

Statt der Funktion MathCos() kann die Funktion [cos\(\)](#) verwendet werden.

## MathExp

Gibt den Wert der Zahl  $e$  in der Potenz  $d$  zurück.

```
double MathExp(  
    double value    //Potenz für die Zahl e  
);
```

### Parameter

*value*

[in] Zahl, die Potenz spezifiziert.

### Rückgabewert

Wert des Typs double. Beim überlauf gibt die Funktion INF (Infinitaet) zurück, beim Ordnungsverlust gibt MathExp 0 zurück.

### Bemerkung

Statt der Funktio MathExp() kann die Funktion [exp\(\)](#) verwendet werden.

### Sehen Sie auch

[Realtypen \(double, float\)](#)



## MathFloor

Gibt den ganzzahligen Wert, der der naechste von unten ist.

```
double MathFloor(  
    double val    // Zahl  
);
```

### Parameter

*val*

[in] Numerischer Wert.

### Rückgabewert

Numerischer Wert, der die maximale Ganzzahl vertritt, die weniger als oder gleich *val* ist.

### Bemerkung

Statt der Funktion MathFloor() kann die Funktion [floor\(\)](#) verwendet werden.

## MathLog

Gibt Napierschen Logarithmus zurück.

```
double MathLog(  
    double val    //Zahl, Logarithmus zu nehmen  
);
```

### Parameter

*val*

[in] Werten, dessen Logarithmus berechnet werden muss.

### Rückgabewert

Natuerlicher Logarithmus val im Erfolgsfall. Wenn val negativ ist, gibt die Funktion NaN (unbestimmten Wert) zurück. Wenn val gleich 0 ist, gibt die Funktion INF (Infinitaet) zurück .

### Bemerkung

Statt der Funktion MathLog() kann die Funktion [log\(\)](#) verwendet werden.

### Sehen Sie auch

[Realtypen \(double, float\)](#)

## MathLog

Gibt Dezimallogarithmus zurück.

```
double MathLog10(  
    double val    // Zahl, Logarithmus zu nehmen  
);
```

### Parameter

*val*

[in] Wert, dessen Dezamallogarithmus berechnet werden muss.

### Rückgabewert

Dezimallogarithmus val im Erfolgsfall. Wenn val negativ ist, gibt die Funktion NaN (unbestimmten Wert) zurück. Wenn val gleich 0 ist, gibt die Funktion INF (Infinitaet) zurück .

### Bemerkung

Statt der Funktion MathLog10() kann die Funktion [log10\(\)](#) verwendet werden.

### Sehen Sie auch

[Realtypen \(double, float\)](#)

## MathMax

Funktion gibt maximalen Wert der zwei numerischen Werte.

```
double MathMax(  
    double value1,    // erster Wert  
    double value2     // zweiter Wert  
);
```

### Parameter

*value1*

[in] Der erste numerische Wert.

*value2*

[in] Der zweite numerische Wert.

### Rückgabewert

Die grösste aus zwei Zahlen.

### Bemerkung

Statt der Funktion `MathMax()` kann die Funktion `fmax()` verwendet werden. Funktionen `fmax()`, `fmin()`, `MathMax()`, `MathMin()` können mit ganzzahligen Typen arbeiten, ohne sie zum Typ `double` zu reduzieren.

Wenn Parameter verschiedener Typen in die Funktion übertragen werden, wird der Parameter des niedrigen Typs automatisch zum höheren Typ reduziert. Typ des Rückgabewertes entspricht dem älteren Typ.

Bei der Übertragung desselben Typs, wird keine Typenreduzierung durchgeführt.

## MathMin

Funktion gibt minimalen Wert der zwei numerischen Werte zurück.

```
double MathMin(  
    double value1,    // erster Wert  
    double value2     // zweiter Wert  
);
```

### Parameter

*value1*

[in] Erster numerischer Wert.

*value2*

[in] Zweiter numerischer Wert.

### Rückgabewert

die kleinste von zwei Zahlen.

### Bemerkung

Statt der Funktion MathMin() kann die Funktion [fmin\(\)](#) verwendet werden. Funktionen [fmax\(\)](#), [fmin\(\)](#), [MathMax\(\)](#), MathMin() können mit ganzzahligen Typen arbeiten, ohne sie zum Typ double zu reduzieren.

Wenn Parameter verschiedener Typen in die Funktion übertragen werden, wird der Parameter des niedrigen Typ automatisch zum Parameter des höheren Typs [reduziert](#). Typ des Rückgabewertes entspricht dem höheren Typ.

Wenn Daten desselben Typs übertragen werden, wird keine Typenreduzierung durchgeführt.

## MathMod

Gibt Realrest vom Dividieren der zwei Zahlen zurück.

```
double MathMod(  
    double value,      // Dividend  
    double value2      // Divisor  
);
```

### Parameter

*value*

[in] Dividend Wert.

*value2*

[in] Divisor Wert.

### Rückgabewert

Funktion MathMod berechnet Realrest  $f$  von  $val / y$  so dass,  $val = i * y + f$ , wo  $i$  eine Ganzzahl ist,  $f$  hat dasselbe Zeichen wie  $val$ , und absoluter Wert  $f$  ist weniger als absoluter Wert von  $y$ .

### Bemerkung

Statt der Funktion MathMod() kann die Funktion [fmod\(\)](#) verwendet werden.

## MathPow

Erhebt die Basis in die angegebene Potenz.

```
double MathPow(  
    double base,           // Basis  
    double exponent        // Potenzwert  
);
```

### Parameter

*base*

[in] Basis.

*exponent*

[in] Potenzwert.

### Rückgabewert

Wert der Basis, die in die angegebene Potenz erhebt ist.

### Bemerkung

Statt der Funktion MathPow() kann die Funktion [pow\(\)](#) verwendet werden.

## MathRand

Gibt pseudozufällige Ganzzahl im Bereich von 0 bis 32767 zurück.

```
int MathRand();
```

### Rückgabewert

Ganzzahl im Bereich von 0 bis 32767.

### Bemerkung

Vor dem zweiten Aufruf der Funktion muss die Funktion [MathSrand](#) verwendet werden, um Generator der pseudozufälligen Zahlen in die Anfangsposition einzustellen.

### Bemerkung

Statt der Funktion MathRand() kann die Funktion [rand\(\)](#) verwendet werden.



## MathRound

Gibt den Wert zurück, der bis zur naechsten Ganzzahl des angegebenen numerischen Wertes gerundet ist.

```
double MathRound(  
    double value    // der zu gerundete Wert  
);
```

### Parameter

*value*

[in] Numerischer Wert für Runden.

### Rückgabewert

Wert, der bis zur naechsten Ganzzahl gerundet ist.

### Bemerkung

Statt der Funktion MathRound() kann die Funktion [round\(\)](#) verwendet werden.

## MathSin

Gibt Sinus des angegebenen Winkels zurück.

```
double MathSin(  
    double value    // Argument in Radianen  
);
```

### Parameter

*value*

[in] Winkel in Radianen.

### Rückgabewert

Sinus des Winkels in Radianen. Gibt Wert im Bereich von -1 bis 1 zurück.

### Bemerkung

Statt der Funktion MathSin() kann die Funktion [sin\(\)](#) verwendet werden.

## MathSqrt

Gibt Quadratwurzel der Zahl zurück.

```
double MathSqrt(  
    double value    // positive Zahl  
);
```

### Parameter

*value*

[in] Positiver numerischer Wert.

### Rückgabewert

Quadratwurzel der Zahl *value*. Wenn *value* negativ ist, gibt MathSqrt den Wert NaN (unbestimmten Wert) zurück.

### Bemerkung

Statt der Funktion MathSqrt() kann die Funktion [sqrt\(\)](#) verwendet werden.

### Sehen Sie auch

[Realtypen \(double, float\)](#)

## MathSrand

Stellt Anfangsposition für Generieren einer Folge der pseudozufälligen Ganzzahlen.

```
void MathSrand(  
    int seed    // initialisierende Zahl  
);
```

### Parameter

*seed*

[in] Anfangszahl für die Reihe der zufälligen Zahlen.

### Rückgabewert

Keinen Rückgabewert.

### Bemerkung

Funktion [MathRand\(\)](#) wird für Erzeugen einer Folge von Pseudo-Zufallszahlen verwendet. Aufruf von MathSrand() mit einer bestimmten Initialisierungszahl ermöglicht es, immer die gleiche Folge von Pseudozufallszahlen zu erhalten.

Zum garantierten Erhalt der nicht wiederkehrenden Abfolge verwenden Sie den Aufruf von MathSrand(GetTickCount()), da der Wert von [GetTickCount\(\)](#) sich ab dem Zeitpunkt des Starts des Betriebssystems erhöht und nicht innerhalb von 49 Tage wiederholt, bis der eingebaute Zähler von Millisekunden überläuft. Verwenden von MathSrand(TimeCurrent()) ist nicht geeignet, weil die Funktion [TimeCurrent\(\)](#) gibt die Zeit der letzte Tick zurück, die für eine lange Zeit unverändert sein kann, zum Beispiel am Wochenende.

Die Initialisierung des Zufallsgenerators mit MathSrand() für Indikatoren und Expert Advisors wird am besten in der OnInit()-Handler getan; es spart Ihnen den folgenden mehrere Neustarts der Generator in OnTick() und OnCalculate()..

Anstelle der Funktion MathSrand(), [srand\(\)](#)-Funktion verwendet werden kann.

### Beispiel:

```

#property description "Der Indikator zeigt den zentralen Grenzwertsatz, der besagt:"
#property description "Die Summe von einer ausreichend großen Anzahl von schwach abh 
#property description "mit ann hernd gleicher Gr  e (keiner der Summanden dominiert,"
#property description "oder macht einen entscheidenden Beitrag zu der Summe), hat eine

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots 1
//--- Eigenschaften der graphischen Konstruktion
#property indicator_label1 "Label"
#property indicator_type1 DRAW_HISTOGRAM
#property indicator_color1 clrRoyalBlue
#property indicator_style1 STYLE_SOLID
#property indicator_width1 5
//--- Variable input
input int sample_number=10;
//--- Ein Indikator-Puffer f r die Erstellung der Verteilung
double LabelBuffer[];
//--- Z hler von Ticks
double ticks_counter;
//+-----+
//| Custom indicator initialization function |
//+-----+
void OnInit()
{
//--- Bindung des Indikator-Puffers mit dem Array
SetIndexBuffer(0,LabelBuffer,INDICATOR_DATA);
//--- Den Indikator-Puffer aus der Gegenwart in die Vergangenheit umwenden
ArraySetAsSeries(LabelBuffer,true);
//--- Den Zufallszahlengenerator initialisieren
MathSrand(GetTickCount());
//--- Den Z hler von ticks initialisieren
ticks_counter=0;
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
//--- Mit dem Null-Z hler setzen wir den Indikator-Puffer zur ck
if(ticks_counter==0) ArrayInitialize(LabelBuffer,0);
//--- Den Z hler erh hen
ticks_counter++;
//--- Wir sollten in regelm  igen Abst nden den Z hler von Ticks zur cksetzen, um die
if(ticks_counter>100)
{
Print("Wir haben die Indikator-Werte zur ckgesetzt, beginnen wir F llen von den
ticks_counter=0;
}
//--- Bekommen Wir einen Auszug von Zufallswerten als die Summe der drei Zahlen von 0
for(int i=0;i<sample_number;i++)
{

```

```
//--- Berechnung des Index der Zelle, wo die Zufallszahl fällt als die Summe der
int rand_index=0;
//--- Bekommen wir drei Zufallszahlen von 0 bis 7
for(int k=0;k<3;k++)
{
    //--- Ein Rest der Division durch 7 wird ein Wert von 0 bis 6 zurückgeben
    rand_index+=MathRand()%7;
}
//--- Erhöhen wir den Wert in der Zelle Nummer rand_index um 1
LabelBuffer[rand_index]++;
}
//--- Verlassen wir den Handler OnCalculate()
return(rates_total);
}
```

## MathTan

Gibt Tangens der Zahl zurück.

```
double MathTan(  
    double rad    // Argument in Radianen  
);
```

### Parameter

*rad*

[in] Winkel in Radianen.

### Rückgabewert

Tangens der Zahl *rad*. Wenn *rad* hoeher als oder gleich 263 ist oder weniger als oder gleich -263 ist, wird der Wert verlorengegangen und die Funktion gibt unbestimmte Zahl zurück.

### Bemerkung

Statt der Funktion `MathTan()` kann die Funktion `tan()` verwendet werden.

### Sehen Sie auch

[Realtypen \(double, float\)](#)

## MathIsValidNumber

Prueft die Richtigkeit der Realzahl

```
bool MathIsValidNumber(  
    double number    // die zu gepruefte Zahl  
);
```

### Parameter

*number*

[in] Gepruefte Zahl.

### Rückgabewert

Gibt true zurück, wenn die gepruefte Zahl eine zulaessige Realzahl ist. Wenn die gepruefte Zahl plus oder minus Infinitaet "Nichtzahl" (NaN - not a number) ist, gibt die Funktion false zurück.

### Beispiel:

```
double abnormal=MathArcsin(2.0);  
if(!MathIsValidNumber(abnormal)) Print("Achtung! MathArcsin(2.0) = ",abnormal);
```

### Sehen Sie auch

[Realtypen \(double, float\)](#)



## Stringfunktionen

Gruppe der Funktionen für Arbeit mit Daten des Typs [string](#).

| Funktion                           | Massnahme                                                                                                                                                                                |
|------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">StringAdd</a>          | Fuegt ein String in den Platz der angegebenen Unterzeile                                                                                                                                 |
| <a href="#">StringBufferLen</a>    | Gibt die Größe des Puffers zurück, der für den String verteilt wurde                                                                                                                     |
| <a href="#">StringCompare</a>      | Vergleicht zwei Strings und gibt 1 zurück, wenn der erste String größer als der zweite ist, 0 - wenn sie gleich sind; -1 (minus eins) - wenn der erste String kleiner als der zweite ist |
| <a href="#">StringConcatenate</a>  | Formiert einen String aus übertragenen Parametern                                                                                                                                        |
| <a href="#">StringFill</a>         | Fuellt den angegebenen String durch die gewählten Symbole aus                                                                                                                            |
| <a href="#">StringFind</a>         | Suche des Teilstrings im String                                                                                                                                                          |
| <a href="#">StringGetCharacter</a> | Gibt den Wert des Symboles, das in der angegebenen Position des Strings liegt                                                                                                            |
| <a href="#">StringInit</a>         | Initialisiert den String durch die gewählten Symbole und stellt die angegebene Stringgrosse bereit                                                                                       |
| <a href="#">StringLen</a>          | Gibt die Zahl der Symbole im String zurück                                                                                                                                               |
| <a href="#">StringReplace</a>      | Ersetzt alle Substrings im String mit einer bestimmten Abfolge von Symbole                                                                                                               |
| <a href="#">StringSetCharacter</a> | Gibt die Kopie des Strings mit dem veränderten Wert des Symboles in der angegebenen Position                                                                                             |
| <a href="#">StringSplit</a>        | Bekommt Teilstrings durch einen angegebenen Trennzeichen aus dem angegebenen String, gibt die Anzahl der Teilstrings zurück                                                              |
| <a href="#">StringSubstr</a>       | Extrahiert die Substring aus dem Textstring, die mit der angegebenen Position beginnt                                                                                                    |
| <a href="#">StringToLower</a>      | Wandelt alle Symbole des angegebenen Strings in kleine Symbole um                                                                                                                        |
| <a href="#">StringToUpper</a>      | Wandelt alle Symbole des angegebenen Strings in grosse Symbole um                                                                                                                        |
| <a href="#">StringTrimLeft</a>     | Entfernt Zeilenvorschub, Spaces und Tabs am Anfang des Strings                                                                                                                           |
| <a href="#">StringTrimRight</a>    | Entfernt Zeilenvorschub, Spaces und Tabs am                                                                                                                                              |

|  |                  |
|--|------------------|
|  | Ende des Strings |
|--|------------------|

## StringAdd

Fuegt die angegebene Unterzeile zum Ende des Strings.

```
bool StringAdd(
    string& string_var,      // String, zu dem zugefuegt wird
    string add_substring    // String, der zugefuegt wird.
);
```

### Parameter

*string\_var*

[in][out] String, zu dem ein anderer ergaenzt wird.

*add\_substring*

[in] String, der am Ende des Ausgangsstrings ergaenzt wird.

### Rückgabewert

Im Erfolgsfall gibt true zurück, anderenfalls false. Für die Erhaltung des Codes des [Fehlers](#) muss die Funktion [GetLastError\(\)](#) aufgerufen werden.

### Beispiel:

```
void OnStart()
{
    long length=1000000;
    string a="a",b="b",c;
    //--- der erste Weg
    uint start=GetTickCount(),stop;
    long i;
    for(i=0;i<length;i++)
    {
        c=a+b;
    }
    stop=GetTickCount();
    Print("time for 'c = a + b' = ",(stop-start)," milliseconds, i = ",i);

    //--- der zweite Weg
    start=GetTickCount();
    for(i=0;i<length;i++)
    {
        StringAdd(a,b);
    }
    stop=GetTickCount();
    Print("time for 'StringAdd(a,b)' = ",(stop-start)," milliseconds, i = ",i);

    //--- der dritte Weg
    start=GetTickCount();
    a="a"; // initialisieren wir die Variable a erneut
    for(i=0;i<length;i++)
```

```
{  
    StringConcatenate(c,a,b);  
};  
stop=GetTickCount();  
Print("time for 'StringConcatenate(c,a,b)' = ",(stop-start)," milliseconds, i = ",i  
;}
```

Sehen Sie auch

[StringConcatenate](#)

## StringBufferLen

Gibt die Puffergrösse zurück, der für den String verteilt wurde.

```
int StringBufferLen(  
    string string_var    // Zeile  
)
```

### Parameter

*string\_var*  
[in] Zeile.

### Rückgabewert

Wert 0 bedeutet, dass der String konstant ist und die Puffergrösse nicht verändert werden kann. -1 bedeutet, dass der String dem Kundenterminal angehört und die Veränderung des Pufferinhalts zu unbestimmten Ergebnissen führen kann.

### Bemerkung

Minimale Größe des Puffers ist 16.

### Beispiel:

```
void OnStart()  
{  
    long length=1000;  
    string a="a",b="b";  
    //---  
    long i;  
    Print("before: StringBufferLen(a) = ",StringBufferLen(a),  
        "   StringLen(a) = ",StringLen(a));  
    for(i=0;i<length;i++)  
    {  
        StringAdd(a,b);  
    };  
    Print("after: StringBufferLen(a) = ",StringBufferLen(a),  
        "   StringLen(a) = ",StringLen(a));  
};
```

### Sehen Sie auch

[StringAdd](#), [StringInit](#), [StringLen](#), [StringFill](#)

## StringCompare

Vergleicht zwei Strings gibt das Ergebnis des Vergleichs als Ganzzahl zurück.

```
int StringCompare(  
    const string& string1,           // erster String des Vergleichs  
    const string& string2,           // zweiter String des Vergleichs  
    bool case_sensitive=true         // Modus der Groß- und Kleinschreibung bein  
);
```

### Parameters

*string1*

[in] Der erste String.

*string2*

[in] Der zweite String.

*case\_sensitive=true*

[in] Modus der Groß- und Kleinschreibung. Wenn es true ist, dann "A">"a". Wenn es false ist, dann "A"="a". Der Standardwert ist true.

### Rückgabewert

- -1 (minus eins), wenn string1<string2
- 0 (null), wenn string1=string2
- 1 (eins), wenn string1>string2

### Bemerkung

Strings werden Zeichen für Zeichen verglichen, Zeichen werden in alphabetischer Reihenfolge nach der aktuellen Codepage verglichen.

### Beispiel:

```
void OnStart()
{
//--- Was mehr ist, einen Apfel oder ein Haus?
    string s1="Apple";
    string s2="home";

//--- vergleichen, Groß/Kleinschreibung beachtet wird
    int result1=StringCompare(s1,s2);
    if(result1>0) PrintFormat("Vergleichen, Groß/Kleinschreibung beachtet wird: %s > %s",s1,s2);
    else
    {
        if(result1<0) PrintFormat("Vergleichen, Groß/Kleinschreibung beachtet wird: %s < %s",s1,s2);
        else PrintFormat("Vergleichen, Groß/Kleinschreibung beachtet wird: %s = %s",s1,s2);
    }

//--- vergleichen, Groß/Kleinschreibung nicht beachtet wird
    int result2=StringCompare(s1,s2,false);
    if(result2>0) PrintFormat("Vergleichen, Groß/Kleinschreibung nicht beachtet wird: %s > %s",s1,s2);
    else
    {
        if(result2<0) PrintFormat("Vergleichen, Groß/Kleinschreibung nicht beachtet wird: %s < %s",s1,s2);
        else PrintFormat("Vergleichen, Groß/Kleinschreibung nicht beachtet wird: %s = %s",s1,s2);
    }
/* Ergebnis:
    Vergleich, wenn Groß/Kleinschreibung beachtet wird: Apple < home
    Vergleich, wenn Groß/Kleinschreibung nicht beachtet wird: Apple < home
*/
}
```

#### Sehen Sie auch

[Typ string](#), [CharToString\(\)](#), [ShortToString\(\)](#), [StringToCharArray\(\)](#), [StringToShortArray\(\)](#), [StringGetCharacter\(\)](#), [Kodeseite Verwenden](#)

## StringConcatenate

Formiert den String aus übertragenen Elementen und gibt die Größe des formierten Strings zurück. Parameter können verschiedener Typen sein. Anzahl der Parameter kann nicht weniger als 2 und nicht mehr als 64 sein.

```
int StringConcatenate(  
    string& string_var,    // String für Formieren  
    void argument1        // der erste Parameter jedes einfachen Typs  
    void argument2        // der zweite Parameter jedes einfachen Typs  
    ...                   // folgender Parameter jedes einfachen Typs  
);
```

### Parameter

*string\_var*

[out] String, der im Ergebnis von Konkatenanz formiert wird.

*argumentN*

[in] Jede Werte, die durch Kommas getrennt werden. Von 2 bis 63 Parameter jedes einfachen Typs.

### Rückgabewert

Gibt die Stringlänge zurück, die durch Konkatenanz von Parameter formiert wurde, die in den Typ string umgewandelt wurden. Parameter werden in die Strings nach denselben Regeln umgewandelt, wie in den Funktionen [Print\(\)](#) und [Comment\(\)](#).

### Sehen Sie auch

[StringAdd](#)



## StringFill

Fuellt den angegebenen String mit den angegebenen Symbolen aus.

```
bool StringFill(
    string&    string_var,      // der zu gefuellte String
    ushort    character        // Symbol, mit dem der String gefuellte wird
);
```

### Parameter

*string\_var*

[in][out] String, der mit dem angegebenen Symbol gefuellte wird.

*character*

[in] Symbol, mit dem der String gefuellte wird.

### Rückgabewert

Im Fall des Erfolges gibt true zurück, andernfalls false. Für Erhaltung des [Fehlerkodes](#) muss man die Funktion [GetLastError\(\)](#) aufrufen.

### Bemerkung

Ausfuellung des Strings bedeutet, dass Symbole in String ohne Zwischenoperationen der Erzeugung des neuen Strings und Kopieren eingesetzt werden. Das erlaubt die Zeit der Arbeit mit dem String in der angegebenen Funktion zu reduzieren.

### Beispiel:

```
void OnStart()
{
    string str;
    StringInit(str,20,'_');
    Print("str = ",str);
    StringFill(str,0);
    Print("str = ",str," : StringBufferLen(str) = ", StringBufferLen(str));
};
// Ergebnis
//   str = _____
//   str =   : StringBufferLen(str) = 20
//
```

### Sehen Sie auch

[StringBufferLen](#), [StringLen](#), [StringInit](#)

## StringFind

Suche des Teilstrings im String.

```
int StringFind(  
    string string_value,      // String, in dem gesucht wird  
    string match_substring,  // was wird gesucht  
    int start_pos=0          // mit welcher Position beginnt die Suche  
);
```

### Parameter

*string\_value*

[in] String in dem Suche durchgeführt wird.

*match\_substring*

[in] Die gesuchte Teilstring.

*start\_pos=0*

[in] Position im String, mit dem die Suche gestartet wird.

### Rückgabewert

Gibt die Positionsnummer im String, mit dem der gesuchte Teilstring beginnt, oder -1, wenn der Teilstring nicht gefunden ist

## StringGetCharacter

Gibt den Wert des Symbols zurück, das sich in der angegebenen Stringposition befindet.

```
ushort StringGetCharacter(  
    string  string_value,    // String  
    int     pos              // Symbolposition im String  
);
```

### Parameter

*string\_value*

[in] Zeile.

*pos*

[in] Symbolposition im STring. Kann von 0 bis [StringLen](#)(text) -1 sein.

### Rückgabewert

Symbolcode oder 0 beim Fehler. Für die Erhaltung des [Fehlerkodes](#) muss die Funktion [GetLastError\(\)](#) aufgerufen werden.

## StringInit

Initialisiert den String durch die angegebenen Symbole und gewährleistet die angegebene Stringgrösse.

```
bool StringInit(
    string&    string_var,      // String für die Initialisierung
    int        new_len=0,       // Erforderliche Stringlänge nach der Initialisierung
    ushort     character=0      // Symbol, mit dem der String gefüllt wird
);
```

### Parameter

*string\_var*

[in][out] String, der initialisiert oder deinitialisiert werden muss.

*new\_len=0*

[in] Stringlänge nach der Initialisierung. Wenn die Größe=0, wird der String deinitialisiert, d.h. Stringpuffer wird verteilt und Pufferadresse wird ausgenullt.

*character=0*

[in] Symbol für Ausfüllung des Strings.

### Rückgabewert

Im Erfolgsfall gibt true zurück, anderenfalls false. Für die Erhaltung des [Fehlercodes](#) muss die Funktion [GetLastError\(\)](#) aufgerufen werden

### Bemerkung

Wenn *character=0* und Größe *new\_len>0*, wird der Stringpuffer verteilt und mit Nullen ausgefüllt. Stringgrösse wird gleich 0 sein, denn der ganze Puffer wird mit Endezeichen ausgefüllt werden.

### Beispiel:

```
void OnStart()
{
    //---
    string str;
    StringInit(str,200,0);
    Print("str = ",str," : StringBufferLen(str) = ",
        StringBufferLen(str)," StringLen(str) = ",StringLen(str));
};
/* Ergebnis
str = : StringBufferLen(str) = 200 StringLen(str) = 0
*/
```

### Sehen Sie auch

[StringBufferLen](#), [StringLen](#)

## StringLen

Gibt die Anzahl der Symbole im String zurück.

```
int StringLen(  
    string string_value    // String  
);
```

### Parameter

*string\_value*

[in] String für die Berechnung der Länge.

### Rückgabewert

Anzahl der Symbole im String ohne Endenull.

## StringReplace

Ersetzt alle Substrings im String mit einer bestimmten Abfolge von Symbolen.

```
int StringReplace(  
    string&      str,           // String, in der Substrings ersetzt werden  
    const string find,         // gewünschter Substring  
    const string replacement    // Substring, der in die gefunden Plätze eingefügt  
);
```

### Parameter

*str*

[in][out] Der String, in dem Sie Ersetzung machen.

*find*

[in] Der gewünschte Substring zu ersetzen.

*replacement*

[in] Substring, der in die gefunden Plätze eingefügt wird.

### Rückgabewert

Anzahl der Ersetzungen im Falle des Erfolgs, im Falle des Fehlers -1. Für die Erhaltung des [Fehlercodes](#) muss die Funktion [GetLastError\(\)](#) aufgerufen werden.

### Bemerkung

Wenn die Funktion erfolgreich ist, aber nichts ersetzt ist (Substring nicht gefunden), gibt es 0 zurück.

Der Fehler kann durch falsche *str* oder *find* Parameter sein (leer oder nicht initialisierter String, sehen sie [StringInit\(\)](#)). Außerdem tritt der Fehler auf, wenn es nicht genug Speicher, um die Ersetzung abzuschließen, gibt.

### Beispiel:

```
string text="The quick brown fox jumped over the lazy dog.";
int replaced=StringReplace(text,"quick","slow");
replaced+=StringReplace(text,"brown","black");
replaced+=StringReplace(text,"fox","bear");
Print("Replaced: ", replaced, ". Result=",text);

// Ergebnis
// Replaced: 3. Result=The slow black bear jumped over the lazy dog.
//
```

### Sehen Sie auch

[StringSetCharacter\(\)](#), [StringSubstr\(\)](#)

## StringSetCharacter

Gibt die Kopie des Strings mit dem veränderten Symbolwert in der angegebenen Position zurück.

```
bool StringSetCharacter(
    string&    string_var,      // String
    int        pos,             // Position
    ushort     character        // Symbol
);
```

### Parameter

*string\_var*

[in][out] String.

*pos*

[in] Symbolposition im String. Kann von 0 bis [StringLen](#)(text) sein.

*character*

[in] Symbolcode Unicode.

### Bemerkung

Wenn Wert pos weniger als [Stringlaenge](#) ist und der Wert des Symbolcodes = 0, wird der String reduziert (aber [Puffergroesse](#), der für den String verteilt wurde, ist unverändert). Stringlaenge wird gleich pos.

Wenn Parameterwert pos ist gleich der Stringlaenge, wird das angegebene Symbol zum Stringende hinzugefügt, und die Stringlaenge wird um 1 vergrößert.

### Beispiel:

```
void OnStart()
{
    string str="0123456789";
    Print("before: str = ",str," ,StringBufferLen(str) = ",
        StringBufferLen(str)," StringLen(str) = ",StringLen(str));
    //--- Fügen wir Nullwert in die Mitte der Zeile
    StringSetCharacter(str,6,0);
    Print(" after: str = ",str," ,StringBufferLen(str) = ",
        StringBufferLen(str)," StringLen(str) =",StringLen(str));
    //--- fügen wir Symbol zum Ende der Zeile hinzu
    int size=StringLen(str);
    StringSetCharacter(str,size,'+');
    Print("addition: str = ",str," ,StringBufferLen(str) = ",
        StringBufferLen(str)," StringLen(str) =",StringLen(str));
};
/* Ergebnis
before: str = 0123456789 ,StringBufferLen(str) = 0 StringLen(str) = 10
after: str = 012345 ,StringBufferLen(str) = 16 StringLen(str) = 6
addition: str = 012345+ ,StringBufferLen(str) = 16 StringLen(str) = 7
*/
```

Sehen Sie auch

[StringBufferLen](#), [StringLen](#), [StringFill](#), [StringInit](#)



## StringSplit

Bekommt Teilstrings durch ein angegebenes Trennzeichen aus dem angegebenen String, gibt die Anzahl der Teilstrings zurück.

```
int StringSplit(
    const string   string_value,      // String in der Teilstrings gesucht werden
    const ushort   separator,         // Trennzeichen
    string         & result[]         // Das per Referenz übergebene Array, um die g
);
```

### Parameter

*string\_value*

[in] Der String, auf den die Teilstrings bekommen werden soll. Der String selbst wird nicht verändert.

*pos*

[in] Code des Trennzeichen. Um der Code zu bekommen, können Sie die Funktion [StringGetCharacter\(\)](#) benutzen.

*result[]*

[out] Ein Array der Strings, das die gefundenen Teilstrings erhält.

### Rückgabewert

Die Anzahl der empfangenen Strings im Array *result[]*. Wenn das Trennzeichen im übergebenen String nicht gefunden wird, dann wird nur der Source-String in das Array Leitung gelegt werden.

Wenn der String *string\_value* leer oder NULL ist, gibt die Funktion null zurück. Im Falle eines Fehlers, gibt die Funktion -1 zurück. Um die Information über [Fehler](#) zu erhalten, muss man die Funktion [GetLastError\(\)](#) aufrufen.

### Beispiel:

```
string to_split="_Leben_ist_schön_";// String der in Teilstrings unterteilt wird
string sep="_";                      // Trennzeichen
ushort u_sep;                        // Code des Trennzeichen
string result[];                     // Array um Teilstrings zu bekommen
//--- Code des Trennzeichen erhalten
u_sep=StringGetCharacter(sep,0);
//--- Den String in Teilstrings unterteilen
int k=StringSplit(to_split,u_sep,result);
//--- Kommentar anzeigen
PrintFormat("Empfangene Teilstrings: %d. Benutzter Trennzeichen '%s' mit Code %d",k,u_sep);
//--- Jetzt anzeigen alle Teilstrings
if(k>0)
{
    for(int i=0;i<k;i++)
    {
        PrintFormat("result[%d]=""%s\"",i,result[i]);
    }
}
```

### Sehen Sie auch

[StringReplace\(\)](#), [StringSubstr\(\)](#), [StringConcatenate\(\)](#)

## StringSubstr

Extrahiert den Teilstring aus der Textzeile, die mit der angegebenen Position beginnt.

```
string StringSubstr(  
    string  string_value,      // String  
    int     start_pos,        // Position für Starten  
    int     length=-1         // Laenge des extrahierten String  
);
```

### Parameter

*string\_value*

[in] String, aus dem der Teilstring extrahiert werden muss.

*start\_pos*

[in] Anfangsposition des Teilstrings. Kann von 0 bis [StringLen](#)(text) -1 sein.

*length=-1*

[in] Laenge des extrahierten Teilstrings. Wenn der Parameterwert -1 ist oder Parameter nicht eingestellt ist, wird der Teilstring von der angegebenen Position bis Stringende extrahiert werden.

### Rückgabewert

Kopie des extrahierten Teilstrings, wenn moeglich. Anderenfalls kehrt der Leerstring zurück.

## StringToLower

Wandelt alle Symbole des angegebenen Strings in kleine um.

```
bool StringToLower(  
    string& string_var    // String für Verarbeitung  
);
```

### Parameter

*string\_var*  
[in][out] String.

### Rückgabewert

Im Erfolgsfall gibt true zurück, anderenfalls false. Für die Erhaltung des [Fehlerkodes](#) muss die Funktion [GetLastError\(\)](#) aufgerufen werden.

## StringToUpper

Wandelt alle Symbole in grosse um.

```
bool StringToUpper(  
    string& string_var    // String für die Verarbeitung  
);
```

### Parameter

*string\_var*

### Rückgabewert

Im Erfolgsfall gibt true zurück, anderenfalls false. Für die Erhaltung des [Fehlercodes](#) muss die Funktion [GetLastError\(\)](#) aufgerufen werden.

## StringTrimLeft

Entfernt Zeilenvorschub, Spaces und Tabs im linken Teil des Strings bis dem ersten bedeutungsvollen Symbol. String wird am Platz modifiziert.

```
int StringTrimLeft(  
    string& string_var    // Zeile für Reduzieren  
);
```

### Parameter

*string\_var*

[in][out] String, der links reduziert werden muss.

### Rückgabewert

Gibt die Anzahl der reduzierten Symbole zurück.

## StringTrimRight

Entfernt Zeilenvorschub, Spaces und Tabs im rechten Teil des Strings nach dem letzten bedeutungsvollen Symbol. String wird am Platz modifiziert.

```
int StringTrimRight(  
    string& string_var    // String für Reduzieren  
);
```

### Parameter

*string\_var*

[in][out] String, der rechts reduziert werden muss.

### Rückgabewert

Gibt die Anzahl der geschnittenen Symbole zurück.

## Datum und Zeit

Gruppe von Funktionen, die Arbeit mit den Daten der Art [datetime](#) (ganze Zahl, als Sekundenzahl seit 1. Januar 1970 0:00gesehen) gewährleistet.

Um die Zähler und hochauflösende Timer zu arrangieren, verwenden Sie die Funktion [GetTickCount\(\)](#), die den Wert in Millisekunden gibt.

| Funktion                            | Massnahme                                                                                                                                   |
|-------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">TimeCurrent</a>         | Bringt den letzten bekannten Serevernamen (Zeit der letzten Kotierung) in datetime Format zurück                                            |
| <a href="#">TimeTradeServer</a>     | Bringt den laufenden Berechnungszeitrum des Handelsservers zurück                                                                           |
| <a href="#">TimeLocal</a>           | Bringt die lokale Computerzeit in datetime Format zurück                                                                                    |
| <a href="#">TimeGMT</a>             | Bringt GMT Zeit in Format datetime mit Rücksicht auf Übergang zur Sommer-/ Winterzeit für lokale Zeit des Computers mit dem Kundenterminals |
| <a href="#">TimeDaylightSavings</a> | Gibt Zeichen des überganges zur Sommer-/ Winterzeit zurück                                                                                  |
| <a href="#">TimeGMTOffset</a>       | Gibt die laufende Differenz zwischen Zeit GMT und lokaler Zeit des Computers in Sekunden mit Rücksicht übergang zur Sommer-/ Winterzeit     |
| <a href="#">TimeToStruct</a>        | Wandelt einen Wert des Typs datetime in eine Variable des Typs der Strukturen MqlDateTime um                                                |
| <a href="#">StructToTime</a>        | Wandelt Variable des Typs der Strukturen MqlDateTime in den Wert des Typs datetime um                                                       |

## TimeCurrent

Gibt den letzten bekannten Wert des Servers, Zeit Des Erhaltes der letzten Quotation für ein der gewählten in "Markt Durchsicht" Symbole. Im Bearbeiter [OnTick\(\)](#) gibt diese Funktion die Zeit des erhaltenen verarbeiteten Ticks. In anderen Fällen (zB, Aufruf in [Bearbeitern](#) OnInit(), OnDeinit(), OnTimer() usw) das ist - [Zeit des Erhaltes der letzten Quotation](#) für jedes Symbol, das in "Markt Durchsicht " zugänglich ist, dieselbe Zeit, die im Titel dieses Fensters gezeigt wird. Zeitwert wird auf dem Handelsserver gebildet und hängt von Zeiteinstellungen im Benutzercomputer nicht. Es gibt zwei Varianten der Funktion.

### Aufruf ohne Parameter

```
datetime TimeCurrent();
```

### Aufruf mit dem Parameter des Typs MqlDateTime

```
datetime TimeCurrent(  
    MqlDateTime& dt_struct    // Variable des Typs Strukturen  
);
```

### Parameter

*dt\_struct*

[out] Variable des Typs Strukturen [MqlDateTime](#).

### Rückgabewert

Wert des Typs [datetime](#)

### Bemerkung

Wenn als Parameter die Variable des Typs der Strukturen MqlDateTime übertragen wurde, wird sie entsprechend ausgefüllt.

Um die Zähler und hochauflösende Timer zu arrangieren, verwenden Sie die Funktion [GetTickCount\(\)](#), die den Wert in Millisekunden gibt.

При работе в тестере стратегий время последней котировки TimeCurrent() моделируется в соответствии с историческими данными.



## TimeTradeServer

Gibt laufende Berechnungszeit des Handelsservers zurück. Zum Unterschied von der Funktion [TimeCurrent\(\)](#), wird die Zeitberechnung im Kundenterminal durchgeführt und hängt von Zeitenstellungen im Benutzercomputer ab. Es gibt 2 Varianten der Funktion.

### Aufruf ohne Parameter

```
datetime TimeTradeServer();
```

### Aufruf mit Parameter des Typs MqlDateTime

```
datetime TimeTradeServer(  
    MqlDateTime& dt_struct    // Variable des Typs Strukturen  
);
```

### Parameter

*dt\_struct*

[out] Variable des Typs Strukturen [MqlDateTime](#).

### Rückgabewert

Wert des Typs [datetime](#)

### Bemerkung

Wenn als Parameter Variable des Typs Strukturen MqlDateTime übertragen wurde, wird sie entsprechend ausgefüllt.

Um die Zähler und hochauflösende Timer zu arrangieren, verwenden Sie die Funktion [GetTickCount\(\)](#), die den Wert in Millisekunden gibt.

При работе в тестере стратегий TimeTradeServer() всегда равно моделируемому серверному времени [TimeCurrent\(\)](#).

## TimeLocal

Gibt lokale Zeit des Computers zurück, auf dem Kundenterminal eingesetzt ist. Es gibt 2 Varianten der Funktion.

### Aufruf ohne Parameter

```
datetime TimeLocal();
```

### Aufruf mit Parameter des Typs MqlDateTime

```
datetime TimeLocal(  
    MqlDateTime& dt_struct    // Variable des Typs Strukturen  
);
```

### Parameter

*dt\_struct*

[out] Variable des Typs Strukturen [MqlDateTime](#).

### Rückgabewert

Wert des Typs [datetime](#)

### Bemerkung

Wenn Variable des Typs Strukturen MqlDateTime als Parameter übertragen wurde, wird sie entsprechend ausgefüllt.

Um die Zähler und hochauflösende Timer zu arrangieren, verwenden Sie die Funktion [GetTickCount\(\)](#), die den Wert in Millisekunden gibt.

При работе в тестере стратегий локальное время TimeLocal() всегда равно моделируемому серверному времени [TimeCurrent\(\)](#).

## TimeGMT

Gibt Zeit GMT zurück, die im Hinblick auf übergang zur Sommer- oder Winterzeit berechnet wird nach der lokalen Zeit des Computers mit dem eingesetzten Kundenterminal wird. Es gibt 2 Varianten der Funktion.

### Aufruf ohne Parameter

```
datetime TimeGMT();
```

### Aufruf mit Parameter des Typs MqlDateTime

```
datetime TimeGMT(  
    MqlDateTime& dt_struct    // Variable des Typs Strukturen  
);
```

### Parameter

*dt\_struct*

[out] Variable des Typs Strukturen [MqlDateTime](#).

### Rückgabewert

Wert des Typs [datetime](#)

### Bemerkung

Wenn als Parameter Variable des Typs Strukturen MqlDateTime übertragen wurde, wird sie entsprechend ausgefüllt.

Um die Zähler und hochauflösende Timer zu arrangieren, verwenden Sie die Funktion [GetTickCount\(\)](#), die den Wert in Millisekunden gibt.

При работе в тестере стратегий время TimeGMT() всегда равно моделируемому серверному времени [TimeTradeServer\(\)](#).

## TimeDaylightSavings

Gibt Korrektur für Sommerzeit in Sekunden zurück, wenn der Übergang zur Sommerzeit stattfand. Hängt von Einstellungen im Benutzercomputer ab.

```
int TimeDaylightSavings();
```

### Rückgabewert

Wenn der Übergang zur Winterzeit (Standardzeit) stattfand, gibt 0 zurück.

## TimeGMTOffset

Gibt die laufende Differenz zwischen der Zeit GMT und lokaler Zeit des Computers in Sekunden im Hinblick auf übergang zur Winter- oder Sommerzeit zurück. Haengt von Zeiteinstellungen im Benutzercomputer ab.

```
int TimeGMTOffset();
```

### Rückgabewert

Wert des Typs int, der den laufenden Unterschied zwischen der lokalen Computerzeit und [Zeit GMT](#) in Sekunden darstellt.

## TimeToStruct

Konvertiert den Wert des Typs `datetime` (Anzahl der Sekunden seit 01.01.1970) in die Variable des Typs Strukturen [MqlDateTime](#).

```
void TimeToStruct (
    datetime      dt,           // Datum und Zeit
    MqlDateTime&  dt_struct     // Struktur für Annahme der Werte
);
```

### Parameter

*dt*

[in] Datumwert für Konvertieren.

*dt\_struct*

[out] Variable des Typs Strukturen `MqlDateTime`.

### Rückgabewert

Keinen Rückgabewert.

## StructToTime

Wandelt die Variable des Typs Strukturen [MqlDateTime](#) in den Wert des Typs [datetime](#) um und gibt den empfangenen Wert zurück.

```
datetime StructToTime(  
    MqlDateTime& dt_struct    // Struktur des Datum und der Zeit  
);
```

### Parameter

*dt\_struct*

[in] Variable des Typs Strukturen MqlDateTime.

### Rückgabewert

Wert des Typs datetime, der Anzahl der Sekunden seit 01.01.1970 enthält.

## Information über das Konto

Funktionen, die Parameter des laufenden Kontos zurückgeben.

| Funktion                                  | Massnahme                                                                                             |
|-------------------------------------------|-------------------------------------------------------------------------------------------------------|
| <a href="#"><u>AccountInfoDouble</u></a>  | Gibt den wert des Typs double der entsprechenden Eigenschaft des Kontos zurück                        |
| <a href="#"><u>AccountInfoInteger</u></a> | Gibt Wert des ganzzahligen Typs (bool,int oder long) der entsprechenden Eigenschaft des Kontos zurück |
| <a href="#"><u>AccountInfoString</u></a>  | Gibt den Wert des Typs string der entsprechenden Eigenschaft des Kontos zurück                        |



## AccountInfoDouble

Gibt den Wert der entsprechenden Eigenschaft des Kontos zurück.

```
double AccountInfoDouble(  
    int property_id    // Identifikator der Eigenschaft  
);
```

### Parameter

*property\_id*

[in] Identifikator der Eigenschaft. Wert kann einer der Werte [ENUM\\_ACCOUNT\\_INFO\\_DOUBLE](#) sein.

### Rückgabewert

Wert des Typs [double](#).

### Beispiel:

```
void OnStart()  
{  
    //--- geben wir die ganze zugängliche Information von der Funktion AccountInfoDouble  
    printf("ACCOUNT_BALANCE = %G",AccountInfoDouble(ACCOUNT_BALANCE));  
    printf("ACCOUNT_CREDIT = %G",AccountInfoDouble(ACCOUNT_CREDIT));  
    printf("ACCOUNT_PROFIT = %G",AccountInfoDouble(ACCOUNT_PROFIT));  
    printf("ACCOUNT_EQUITY = %G",AccountInfoDouble(ACCOUNT_EQUITY));  
    printf("ACCOUNT_MARGIN = %G",AccountInfoDouble(ACCOUNT_MARGIN));  
    printf("ACCOUNT_FREEMARGIN = %G",AccountInfoDouble(ACCOUNT_FREEMARGIN));  
    printf("ACCOUNT_MARGIN_LEVEL = %G",AccountInfoDouble(ACCOUNT_MARGIN_LEVEL));  
    printf("ACCOUNT_MARGIN_SO_CALL = %G",AccountInfoDouble(ACCOUNT_MARGIN_SO_CALL));  
    printf("ACCOUNT_MARGIN_SO_SO = %G",AccountInfoDouble(ACCOUNT_MARGIN_SO_SO));  
};
```

### Sehen Sie auch

[SymbolInfoDouble](#), [SymbolInfoString](#), [SymbolInfoInteger](#), [PrintFormat](#)

## AccountInfoInteger

Gibt den Wert der entsprechenden Eigenschaft des Kontos zurück.

```
long AccountInfoInteger(
    int property_id // Identifikator der Eigenschaft
);
```

### Parameter

*property\_id*

[in] Identifikator der Eigenschaft. Wert kann einer der Werte [ENUM\\_ACCOUNT\\_INFO\\_INTEGER](#) sein.

### Rückgabewert

Wert des Typs [long](#).

### Bemerkung

Eigenschaft muss eine der Typen [bool](#), [int](#) oder [long](#) sein.

### Beispiel:

```
void OnStart()
{
    //--- Geben wir die ganze zugängliche Information von der Funktion AccountInfoInteger
    printf("ACCOUNT_LOGIN = %d", AccountInfoInteger(ACCOUNT_LOGIN));
    printf("ACCOUNT_LEVERAGE = %d", AccountInfoInteger(ACCOUNT_LEVERAGE));
    bool thisAccountTradeAllowed=AccountInfoInteger(ACCOUNT_TRADE_ALLOWED);
    bool EATradeAllowed=AccountInfoInteger(ACCOUNT_TRADE_EXPERT);
    ENUM_ACCOUNT_TRADE_MODE tradeMode=(ENUM_ACCOUNT_TRADE_MODE) AccountInfoInteger(ACCOUNT_TRADE_MODE);
    ENUM_ACCOUNT_STOPOUT_MODE stopOutMode=(ENUM_ACCOUNT_STOPOUT_MODE) AccountInfoInteger(ACCOUNT_STOPOUT_MODE);

    //--- informieren wir über die Möglichkeit, Handelsoperationen durchzuführen
    if(thisAccountTradeAllowed)
        Print("Handel für dieses Konto ist erlaubt");
    else
        Print("Handel für dieses Konto ist erlaubt!");

    //--- stellen wir fest - ob es möglich ist, in diesem Konto mit Experten zu handeln
    Print("Handel durch Ratgeber ist für dieses Konto erlaubt");
    else
        Print("Handel durch Ratgeber ist für dieses Konto verboten!");

    //--- stellen wir Kontotyp fest
    switch(tradeMode)
    {
        case(ACCOUNT_TRADE_MODE_DEMO):
            Print("Das ist ein Demo-Konto");
            break;
        case(ACCOUNT_TRADE_MODE_CONTEST):
```

```
        Print("Das ist ein Konkurrenzkonto");
        break;
    default:Print("Das ist ein tatsaechliches Konto!");
    ;}

//--- stellen wir Mode der Leveleinstellung fest
switch(stopOutMode)
{
    case (ACCOUNT_STOPOUT_MODE_PERCENT):
        Print("Level StopOut wird in Prozenten angegeben");
        break;
    default:Print("Level StopOut wird im Geldausdruck angegeben");
    ;}
;}
```

Sehen Sie auch

[Kontoinformation](#)

## AccountInfoString

Gibt den Wert der entsprechenden Eigenschaft des Kontos zurück.

```
string AccountInfoString(  
    int property_id    // Identifikator der Eigenschaft  
);
```

### Parameter

*property\_id*

[in] Identifikator der Eigenschaft. Wert kann einer der Werte [ENUM\\_ACCOUNT\\_INFO\\_STRING](#) sein.

### Rückgabewert

Wert des Typs [string](#).

### Beispiel:

```
void OnStart()  
{  
    //--- geben wird die ganze zugängliche Information von der Funktion AccountInfoString  
    Print("Name des Brokers = ",AccountInfoString(ACCOUNT_COMPANY));  
    Print("Depositenwaehrung = ",AccountInfoString(ACCOUNT_CURRENCY));  
    Print("Name des Kunden = ",AccountInfoString(ACCOUNT_NAME));  
    Print("Name des Handelsservers = ",AccountInfoString(ACCOUNT_SERVER));  
};
```

### Sehen Sie auch

[Kontoinformation](#)

## Zustandsprüfung

Funktionen, die Parameter des laufenden Zustands des Kundenterminals zurückgeben

| Funktion                                   | Massname                                                                                                         |
|--------------------------------------------|------------------------------------------------------------------------------------------------------------------|
| <u><a href="#">GetLastError</a></u>        | Gibt den wert des letzten Fehlers zurück                                                                         |
| <u><a href="#">IsStopped</a></u>           | Gibt true zurück, wenn mql5-Programm befielt, das Programm zu beenden                                            |
| <u><a href="#">UninitializeReason</a></u>  | Gibt den Kode des Deinitialisierungsgrundes                                                                      |
| <u><a href="#">TerminalInfoInteger</a></u> | Gibt den Wert des ganzzahligen Types der entsprechenden Eigenschaft der Umwelt des mql5-Programms zurück         |
| <u><a href="#">TerminalInfoDouble</a></u>  | Возвращает значение типа double соответствующего свойства окружения mql5-программы                               |
| <u><a href="#">TerminalInfoString</a></u>  | Gibt den Wert desw Typs string der entsprechenden Eigenschaft der Umwelt des mql5-Programms zurück               |
| <u><a href="#">MQLInfoInteger</a></u>      | Gibt den Wert des ganzzahligen Typs der entsprechenden Eigenschaft des eingesetzten mql5-Programms zurück        |
| <u><a href="#">MQLInfoString</a></u>       | Gibt den Wert des Typs string der entsprechenden Eigenschaft des eingesetzten mql5-Programms                     |
| <u><a href="#">Symbol</a></u>              | Gibt Symbolname des laufenden Charts.                                                                            |
| <u><a href="#">Period</a></u>              | Gibt den Wert von Timeframe des laufenden Charts zurück                                                          |
| <u><a href="#">Digits</a></u>              | Gibt die Anzahl der Dezimalzeichen, die Genauigkeit der Berechnung des Preiswertes des laufenden Charts bestimmt |
| <u><a href="#">Point</a></u>               | Gibt den Wert des Punktes des laufenden Instrumentes in der Quotationswaehrung zurück.                           |

## GetLastError

Gibt Inhalt der Systemvariable [\\_LastError](#) zurück.

```
int GetLastError();
```

### Rückgabewert

Gibt Wert des letzten [Fehlers](#), der während der Durchführung des mql5-Programms entstanden ist, zurück.

### Bemerkung

Nach Aufruf der Funktion wird Inhalt der Variable `_LastError` nicht ausgenullt. Für Ausnullen dieser Variable muss die Funktio [ResetLastError\(\)](#) aufgerufen werden.

### Sehen Sie auch

[Rückgabecodes des Handelsservers](#)

## IsStopped

Prüft Zwangsbeenden des mql5-Programms.

```
bool IsStopped();
```

### Rückgabewert

Gibt true zurück, wenn es in der Systemvariable [\\_StopFlag](#) den Wert gibt, der sich von 0 unterscheidet. Der Nicht-Nullwert wird in die Variable `_StopFlag` geschrieben, wenn mql5-Programm befiehlt, Operation zu beenden. In diesem Fall muss man sofort das Programm beenden, anderenfalls wird das Programm von aussen zwangsläufig in 3 Sekunden beendet.

## UninitializeReason

Gibt Kode des [Deinitialisierungsgrundes](#) zurück.

```
int UninitializeReason();
```

### Rückgabewert

Gibt den Wert der Variable [\\_UninitReason](#) zurück, der vor dem Aufruf der Funktion [OnDeinit\(\)](#) formiert wird. Wert hängt vom Grund ab, der zur Deinitialisierung geführt hat.



## TerminalInfoInteger

Gibt Wert der entsprechenden Eigenschaft der Umwelt des mql5-Programms zurück.

```
int TerminalInfoInteger(  
    int property_id    // Identifikator der Eigenschaft  
);
```

### Parameter

*property\_id*

[in] Identifikator der Eigenschaft. Kann einer der Aufzählungswerte [ENUM\\_TERMINAL\\_INFO\\_INTEGER](#) sein.

### Rückgabewert

Wert des Typs int.

## TerminalInfoDouble

Возвращает значение соответствующего свойства окружения mql4-программы.

```
double TerminalInfoDouble(  
    int property_id    // идентификатор свойства  
);
```

### Параметры

*property\_id*

[in] Идентификатор свойства. Может быть одним из значений перечисления [ENUM\\_TERMINAL\\_INFO\\_DOUBLE](#).

### Возвращаемое значение

Значение типа double.

## TerminalInfoString

Funktion gibt den Wert der entsprechenden Eigenschaft der Umwelt des mql5-Programms zurück. Eigenschaft muss des Typs string sein.

```
string TerminalInfoString(  
    int property_id    // Identifikator der Eigenschaft  
);
```

### Parameter

*property\_id*

[in] Identifikator der Eigenschaft. Kann einer der Aufzählungswerte [ENUM\\_TERMINAL\\_INFO\\_STRING](#) sein.

### Rückgabewert

Wert des Typs string.

## MQLInfoInteger

Gibt den Wert der entsprechenden Eigenschaft des eingesetzten MQL5-Programms zurück.

```
int MQLInfoInteger(  
    int property_id    // Identifikator der Eigenschaft  
);
```

### Parameter

*property\_id*

[in] Identifikator der Eigenschaft. Kann einer der Aufzählungswerte [ENUM\\_MQL\\_INFO\\_INTEGER](#) sein.

### Rückgabewert

Wert des Typs int.

## MQLInfoString

Gibt den Wert der entsprechenden Eigenschaft des eingesetzten MQL5-Programms zurück.

```
string MQLInfoString(  
    int property_id    // Identifikator der Eigenschaft  
);
```

### Parameter

*property\_id*

[in] Identifikator der Eigenschaft. Kann einer der Aufzählungswerte [ENUM\\_MQL\\_INFO\\_STRING](#) sein.

### Rückgabewert

Wert des Typs string.

## Symbol

Gibt Symbolname des laufenden Charts zurück.

```
string Symbol();
```

### Rückgabewert

Wert der Systemvariable Symbol, in der der Symbolname des laufenden Charts aufbewahrt wird.

## Period

Gibt den Wert von Timeframe des laufenden Charts zurück.

```
ENUM_TIMEFRAMES Period();
```

### Rückgabewert

Inhalt der Variable [\\_Period](#), in Timeframewert des laufenden Charts aufbewahren wird. Wert kann einer der Aufzählungswerte [ENUM\\_TIMEFRAMES](#) sein.

### Sehen Sie auch

[PeriodSeconds](#), [Chartperioden](#), [Datum und Zeit](#), [Objektsichtbarkeit](#)

## Digits

Gibt die Anzahl der Dezimalzeichen, die die Genauigkeit der Preisberechnung des Symbols des laufenden Charts bestimmt.

```
int Digits();
```

### Rückgabewert

Wert der Variable [\\_Digits](#), in der die Anzahl der Dezimalzeichen aufbewahren wird, die die Genauigkeit der Preisberechnung des Symbols des laufenden Charts bestimmt.



## Point

Gibt die Punktgrösse des laufenden Symbols in der Quotationswährung zurück.

```
double Point();
```

### Rückgabewert

Wert der Variable [\\_Point](#), in der die Punktgrösse des laufenden Symbols in der Quotationswährung aufbewahrt wird.

## Marktinformation erhalten

Funktionen, um Informationen über Marktstand zu erhalten.

| Funktion                                      | Massnahme                                                                                                                                      |
|-----------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#"><u>SymbolsTotal</u></a>           | Gibt die Anzahl zugaenglicher (gewaehlter in MarketWatch oder aller) Symbole zurück                                                            |
| <a href="#"><u>SymbolName</u></a>             | Gibt den Namen des bezeichneten Symbols zurück                                                                                                 |
| <a href="#"><u>SymbolSelect</u></a>           | Waehlt das Symbol im Fenster MarketWatch oder entfernt das Symbol aus dem Fenster                                                              |
| <a href="#"><u>SymbolsSynchronized</u></a>    | Überprüft ob die Daten des angegebenen Symbol mit Daten auf dem Handelsservers <a href="#"><u>synchronisiert</u></a> sind                      |
| <a href="#"><u>SymbolInfoDouble</u></a>       | Gibt den Wert des Typs double des angegebenen Symbols für die entsprechende Eigenschaft zurück                                                 |
| <a href="#"><u>SymbolInfoInteger</u></a>      | Gibt den Wert des ganzzahligen Typs (long, datetime, int oder bool) des angegebenen Symbols für die entsprechende Eigenschaft zurück           |
| <a href="#"><u>SymbolInfoString</u></a>       | Gibt den Wert des Typs string des angegebenen Symbols für die entsprechende Eigenschaft zurück                                                 |
| <a href="#"><u>SymbolInfoMarginRate</u></a>   | Возвращает коэффициенты взимания маржи в зависимости от типа и направления ордера                                                              |
| <a href="#"><u>SymbolInfoTick</u></a>         | Gibt laufende Preise für das angegebene Symbol in der Variable des Typs <a href="#"><u>MqlTick</u></a>                                         |
| <a href="#"><u>SymbolInfoSessionQuote</u></a> | Ermöglicht die Anfangszeit und die Abschlusszeit der angegebenen Kotierungssession für das angegebene Symbol und für den angegebenen Wochentag |
| <a href="#"><u>SymbolInfoSessionTrade</u></a> | Ermöglicht die Anfangszeit und die Abschlusszeit der angegebenen Handelssession für das angegebene Symbol und für den angegebenen Wochentag    |
| <a href="#"><u>MarketBookAdd</u></a>          | Sorgt für Eroeffnung von DOM für das gewaehlte Symbol, und subskribiert auf Nachrichten über DOM Veränderung                                   |
| <a href="#"><u>MarketBookRelease</u></a>      | Sorgt für Schliessen von DOM für das gewaehlte Symbol und annulliert Subskription auf Nachrichten über DOM Veränderung                         |

[MarketBookGet](#)

Gibt Feld des Typs Strukturen [MqlBookInfo](#), das Aufzeichnungen von DOM des angegebenen Symbols enthält

## SymbolsTotal

Gibt die Anzahl der zugänglichen Symbole (gewählten in MarketWatch oder allen).

```
int SymbolsTotal(  
    bool selected    // true - nur Symbole in MarketWatch  
);
```

### Parameter

*selected*

[in] Mode der Anforderung. Kann Werte true oder false haben.

### Rückgabewert

Wenn Parameter *selected* gleich true ist, gibt die Funktion die Anzahl der gewählten Symbole in MarketWatch zurück. Wenn der Wert false ist, gibt die Funktion die ganze Anzahl aller Symbole zurück.

## SymbolName

Gibt Name des angegebenen Symbols zurück.

```
string SymbolName(  
    int    pos,           // Nummer in der Liste  
    bool   selected       // true - nur Symbole in MarketWatch  
);
```

### Parameter

*pos*

[in] Symbolnummer der Ordnung nach

*selected*

[in] Mode der Anforderung. Wenn der Wert true ist, wird das Symbol aus der Liste der in MarketWatch gewählten Symbole genommen. Wenn der Wert false ist, wird das Symbol aus der allgemeinen Liste genommen.

### Rückgabewert

Wert des Typs string mit dem Namen des Symbols.

## SymbolSelect

Wählt das Symbol im Fenster MarketWatch oder entfernt das Symbol aus dem Fenster.

```
bool SymbolSelect(  
    string name,           // Symbolname  
    bool select            // hinzufügen oder entfernen  
);
```

### Parameter

*name*

[in] Symbolname.

*select*

[in] Schalter. Wenn der Wert false ist, muss das Symbol aus dem Fenster MarketWatch entfernt werden, anderenfalls muss das Symbol im Fenster MarketWatch ausgewählt werden. Symbol kann nicht entfernt werden, wenn es offene Charts mit diesem Symbol oder offene Positionen für dieses Symbol gibt.

### Rückgabewert

Im Fall des Misserfolges gibt die Funktion false zurück.

## SymbolIsSynchronized

Prueft, ob Daten des angegebenen Symbols im Terminal mit Daten im Handelsserver synchronisiert werden oder nicht

```
bool SymbolIsSynchronized(  
    string name,          // Symbolname  
);
```

### Parameter

*name*

[in] Symbolname.

### Rückgabewert

Wenn Daten [synchronisiert](#) sind, wird true zurückgegeben, anderenfalls false.

### Sehen Sie auch

[SymbolInfoInteger](#), [Datenzugang organisieren](#)

## SymbolInfoDouble

Gibt die entsprechende Eigenschaft des angegebenen Symbols zurück. Es gibt zwei Varianten der Funktion.

1. gibt den Wert der Eigenschaft sofort zurück.

```
double SymbolInfoDouble(
    string          name,      // Symbol
    ENUM_SYMBOL_INFO_DOUBLE prop_id // Identifikator der Eigenschaft
);
```

2. Gibt true oder false zurück, abhängig davon, ob die Funktion erfolgreich durchgeführt wird. Im Erfolgsfall wird der Wert der Eigenschaft in die Empfangsvariable gesetzt, die durch Referenz vom letzten Parameter übertragen wird.

```
bool SymbolInfoDouble(
    string          name,      // Symbol
    ENUM_SYMBOL_INFO_DOUBLE prop_id, // Identifikator der Eigenschaft
    double&         double_var // hier nehmen wir den wert der Eigenschaft
);
```

### Parameter

*name*

[in] Symbolname.

*prop\_id*

[in] Identifikator der Symboleigenschaft. Wert kann einer der Aufzählungswerte [ENUM\\_SYMBOL\\_INFO\\_DOUBLE](#) sein.

*double\_var*

[out] Variable des Typs double, die den Wert der angeforderten Eigenschaft annimmt.

### Rückgabewert

Wert des Typs double. Im Falle der Nichtausführung können Sie [Fehlerinformation](#) durch die Funktion [GetLastError\(\)](#) erhalten:

- 5040 - ungültiger string-Parameter für die Angabe des Symbolnamens,
- 4301 - unbekanntes Symbol (Finanzinstrument)
- 4302 - das Symbol nicht in der "Market Watch" ausgewählt (ist in der Liste nicht verfügbar),
- 4303 - ungültiger Identifikator der Symboleigenschaft.

### Hinweis

Wenn die Funktion wird verwendet, um Informationen über den letzten Tick zu bekommen, ist es besser [SymbolInfoTick\(\)](#) zu verwenden. Es ist möglich, dass seit Verbinden des Terminals an das Handelskonto keine Preise des Symbols empfangen worden. In einem solchen Fall wird der Wert nicht bestimmt.

In den meisten Fällen reicht es aus, [SymbolInfoTick\(\)](#) zu verwenden. Mit dieser Funktion können Sie durch einen einzelnen Anruf Werte von Ask, Bid, Last, Volume und Zeit der des letzten Ticks zu bekommen.



## Beispiel:

```
void OnTick()
{
//--- erhalten wir Spread aus der Symboleigenschaften
    bool spreadfloat=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD_FLOAT);
    string comm=StringFormat("Spread %s = %I64d Punkte\r\n",
                             spreadfloat?"fliessend":"fixiert",
                             SymbolInfoInteger(Symbol(),SYMBOL_SPREAD));
//--- berechnen wir nun Spread selbsstaendig
    double ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
    double bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
    double spread=ask-bid;
    int spread_points=(int)MathRound(spread/SymbolInfoDouble(Symbol(),SYMBOL_POINT));
    comm=comm+"Berechneter Spread = "+(string)spread_points+" Punkte";
    Comment(comm);
};}
```

## SymbolInfoInteger

Gibt die entsprechende Eigenschaft des angegebenen Symbols zurück. Es gibt 2 Varianten der Funktion.

1. Gibt den Eigenschaftswert sofort zurück.

```
long SymbolInfoInteger(
    string          name,      // Symbol
    ENUM_SYMBOL_INFO_INTEGER prop_id // Identifikator der Eigenschaft
);
```

2. Gibt true oder false zurück abhängig von der erfolgreichen Durchführung der Funktion. Im Erfolgsfall wird der Eigenschaftswert in die Empfangsvariable gestellt, die durch Referenz vom letzten Parameter übertragen wird.

```
bool SymbolInfoInteger(
    string          name,      // Symbol
    ENUM_SYMBOL_INFO_INTEGER prop_id, // Eigenschaften des Identifikatoren
    long&          long_var // hier wird der Eigenschaftswert angenommen
);
```

### Parameter

*name*

[in] Symbolname.

*prop\_id*

[in] Identifikator der Symboleigenschaft. Wert kann einer der Aufzählungswerte [ENUM\\_SYMBOL\\_INFO\\_INTEGER](#) sein.

*long\_var*

[out] Variable des Typs long, die den Wert der angeforderten Eigenschaft annimmt.

### Rückgabewert

Wert des Typs long. Im Falle der Nichtausführung können Sie [Fehlerinformation](#) durch die Funktion [GetLastError\(\)](#) erhalten:

- 5040 - ungültiger string-Parameter für die Angabe des Symbolnamens,
- 4301 - unbekanntes Symbol (Finanzinstrument)
- 4302 - das Symbol nicht in der "Market Watch" ausgewählt (ist in der Liste nicht verfügbar),
- 4303 - ungültiger Identifikator der Symboleigenschaft.

### Hinweis

Wenn die Funktion wird verwendet, um Informationen über den letzten Tick zu bekommen, ist es besser [SymbolInfoTick\(\)](#) zu verwenden. Es ist möglich, dass seit Verbinden des Terminals an das Handelskonto keine Preise des Symbols empfangen worden. In einem solchen Fall wird der Wert nicht bestimmt.

In den meisten Fällen reicht es aus, [SymbolInfoTick\(\)](#) zu verwenden. Mit dieser Funktion können Sie durch einen einzelnen Anruf Werte von Ask, Bid, Last, Volume und Zeit der des letzten Ticks zu bekommen.

## Beispiel:

```
void OnTick()
{
    ///--- erhalten wir Spread aus Symboleigenschaften
    bool spreadfloat=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD_FLOAT);
    string comm=StringFormat("Spread %s = %I64d Punkte\r\n",
                             spreadfloat?"fliessend":"fixiert",
                             SymbolInfoInteger(Symbol(),SYMBOL_SPREAD));
    ///--- berechnen wir Spread selbststaendig
    double ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
    double bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
    double spread=ask-bid;
    int spread_points=(int)MathRound(spread/SymbolInfoDouble(Symbol(),SYMBOL_POINT));
    comm=comm+"Berechneter Spread = "+(string)spread_points+" Punkte";
    Comment(comm);
};
```

## SymbolInfoString

Gibt die entsprechende Eigenschaft des angegebenen Symbols zurück. Es gibt 2 Varianten der Funktion.

1. Gibt den Eigenschaftswert sofort zurück.

```
string SymbolInfoString(
    string          name,          // Symbol
    ENUM_SYMBOL_INFO_STRING prop_id // Identifikator der Eigenschaft
);
```

2. Gibt true oder false zurück, abhängig davon, ob die Funktion erfolgreich durchgeführt wurde. Im Erfolgsfall wird der Eigenschaftswert in die Empfangsvariable gestellt, die durch Referenz vom letzten Parameter übertragen wird.

```
bool SymbolInfoString(
    string          name,          // Symbol
    ENUM_SYMBOL_INFO_STRING prop_id, // Identifikator der Eigenschaft
    string&         string_var    // hier wird der Eigenschaftswert angenommen
);
```

### Parameter

*name*

[in] Symbolname.

*prop\_id*

[in] Identifikator der Symboleigenschaft. Wert kann einer der Aufzählungswerte [ENUM\\_SYMBOL\\_INFO\\_STRING](#) sein .

*string\_var*

[out] Variable des Typs string, die den Wert der angeforderten Eigenschaft annimmt .

### Rückgabewert

Wert des Typs string. Im Falle der Nichtausführung können Sie [Fehlerinformation](#) durch die Funktion [GetLastError\(\)](#) erhalten:

- 5040 - ungültiger string-Parameter für die Angabe des Symbolnamens,
- 4301 - unbekanntes Symbol (Finanzinstrument)
- 4302 - das Symbol nicht in der "Market Watch" ausgewählt (ist in der Liste nicht verfügbar),
- 4303 - ungültiger Identifikator der Symboleigenschaft.

### Hinweis

Wenn die Funktion wird verwendet, um Informationen über den letzten Tick zu bekommen, ist es besser [SymbolInfoTick\(\)](#) zu verwenden. Es ist möglich, dass seit Verbinden des Terminals an das Handelskonto keine Preise des Symbols empfangen worden. In einem solchen Fall wird der Wert nicht bestimmt.

In den meisten Fällen reicht es aus, [SymbolInfoTick\(\)](#) zu verwenden. Mit dieser Funktion können Sie durch einen einzelnen Anruf Werte von Ask, Bid, Last, Volume und Zeit der des letzten Ticks zu bekommen.



## SymbolInfoMarginRate

Возвращает коэффициенты взимания маржи в зависимости от типа и направления ордера.

```
bool SymbolInfoMarginRate (
    string          name,                // символ
    ENUM_ORDER_TYPE order_type,         // тип ордера
    double&         initial_margin_rate, // коэффициент взимания начальной маржи
    double&         maintenance_margin_rate // коэффициент взимания поддерживающей маржи
);
```

### Параметры

*name*

[in] Имя символа.

*order\_type*

[in] Тип ордера.

*initial\_margin\_rate*

[in] Переменная типа [double](#) для получения коэффициента взимания начальной маржи. Начальная маржа - это размер гарантийной суммы под совершение сделки объемом в 1 лот соответствующего направления. Умножая коэффициент на начальную маржу, мы можем получить размер средств, который будет зарезервирован на счете при размещении ордера указанного типа.

*maintenance\_margin\_rate*

[out] Переменная типа [double](#) для получения коэффициента взимания поддерживающей маржи. Поддерживающая маржа - это размер минимальной суммы для поддержания открытой позиции объемом в 1 лот соответствующего направления. Умножая коэффициент на поддерживающую маржу, мы можем получить размер средств, который будет зарезервирован на счете после срабатывания ордера указанного типа.

### Возвращаемое значение

Возвращает true в случае удачного выполнения запроса свойств, иначе false.

## SymbolInfoTick

Gibt laufende Preise für das angegebene Symbol in der Variable des Typs MqlTick zurück.

```
bool SymbolInfoTick(  
    string      symbol,      // Symbol  
    MqlTick&    tick         // Strukturreferenz  
);
```

### Parameter

*symbol*

[in] Symbolname.

*tick*

[out] Verweisung auf die Struktur des Typs [MqlTick](#), in die die laufenden Preise und Zeit des letzten Preisenupdates gesetzt werden.

### Rückgabewert

Gibt true im Erfolgsfall zurück, anderenfalls false.

## SymbolInfoSessionQuote

Ermöglicht die Anfangs- und die Abschlusszeit der angegebenen Kotierungssession für das angegebene Symbol und für den angegebenen Wochentag zu erhalten.

```
bool SymbolInfoSessionQuote(  
    string          name,           // Symbolname  
    ENUM_DAY_OF_WEEK day_of_week,   // Wochentag  
    uint           session_index,    // Nummer der Session  
    datetime&       from,           // Anfangszeit der Session  
    datetime&       to              // Abschlusszeit der Session  
);
```

### Parameter

*name*

[in] Symbolname.

*ENUM\_DAY\_OF\_WEEK*

[in] Wochentag, Wert aus der Aufzählung [ENUM\\_DAY\\_OF\\_WEEK](#).

*uint*

[in] Ordnungsnummer der Session, für die Anfangszeit und Abschlusszeit zu erhalten sind. Indizieren der Sessionen fängt mit 0 an.

*from*

[out] Anfangszeit der Session in Sekunden von 00 Stunden 00 Minuten, im erhaltenen Wert muss das Datum ignoriert werden.

*to*

[out] Abschlusszeit der Session in Sekunden von 00 Stunden 00 Minuten, im erhaltenen Wert muss das Datum ignoriert werden.

### Rückgabewert

Wenn die Daten für die angegebenen Session, Symbol und Wochentag erhalten worden sind, gibt es true zurück, anderenfalls false.

### Sehen Sie auch

[Information über das Symbol](#), [TimeToStruct](#), [Datumstruktur](#)



## SymbolInfoSessionTrade

Ermöglicht die Anfangs- und die Abschlusszeit der angegebenen Handelssession für das angegebene Symbol und für den angegebenen Wochentag zu erhalten.

```
bool SymbolInfoSessionTrade(  
    string          name,           // Symbolname  
    ENUM_DAY_OF_WEEK day_of_week,   // Wochentag  
    uint            session_index,   // Nummer der Session  
    datetime&       from,           // Anfangszeit der Session  
    datetime&       to              // Abschlusszeit der Session  
);
```

### Parameter

*name*

[in] Symbolname.

*ENUM\_DAY\_OF\_WEEK*

[in] Wochentag, Wert aus der Aufzählung [ENUM\\_DAY\\_OF\\_WEEK](#).

*uint*

[in] Ordnungsnummer der Session, für die Anfangszeit und Abschlusszeit zu erhalten sind. Indizieren der Sessionen fängt mit 0 an.

*from*

[out] Anfangszeit der Session in Sekunden von 00 Stunden 00 Minuten, im erhaltenen Wert muss das Datum ignoriert werden.

*to*

[out] Abschlusszeit der Session in Sekunden von 00 Stunden 00 Minuten, im erhaltenen Wert muss das Datum ignoriert werden.

### Rückgabewert

Wenn die Daten für die angegebenen Session, Symbol und Wochentag erhalten worden sind, gibt es true zurück, anderenfalls false.

### Sehen Sie auch

[Information über das Instrument](#), [TimeToStruct](#), [Datumstruktur](#)

## MarketBookAdd

Gewährleistet die DOM Eröffnung für das gewählte Symbol und subskribiert auf die Erhaltung der Nachrichten über DOM Veränderung.

```
bool MarketBookAdd(  
    string symbol    // Symbol  
);
```

### Parameter

*symbol*

[in] Symbolname, dessen DOM in dem angegebenen Experten oder Script benutzt werden muss.

### Rückgabewert

Wert true im Fall der erfolgreichen Eröffnung, anderenfalls false.

### Bemerkung

In der Regel muss diese Funktion aus der Funktion [OnInit\(\)](#) oder im Klassenkonstrukteur aufgerufen werden. Für Verarbeitung der Eingangsmeldungen muss die Funktion void [OnBookEvent](#)(string& symbol) vorhanden sein.

### Sehen Sie auch

[DOM Struktur](#) , [Strukturen und Klassen](#)

## MarketBookRelease

Gewährleistet DOM Schliessen für das gewählte Symbol und annulliert Subskription auf die Erhaltung von Meldungen über DOM Veränderung.

```
bool MarketBookRelease(  
    string symbol    // Symbolname  
);
```

### Parameter

*symbol*

[in] Symbolname.

### Rückgabewert

Wert true beim erfolgreichen Schliessen, anderenfalls false.

### Bemerkung

Normalerweise muss diese Funktion aus der Funktion [OnDeinit\(\)](#) aufgerufen werden, wenn in der Funktion [OnInit\(\)](#) die entsprechende Funktion [MarketBookAdd\(\)](#) aufgerufen wurde. Oder sie muss vom Destruktor der Klasse aufgerufen werden, wenn die entsprechende Funktion MarketBookAdd() im Konstruktor dieser Klasse aufgerufen wird.

### Sehen Sie auch

[DOM Struktur](#), [Strukturen und Klassen](#)

## MarketBookGet

Gibt Feld der Strukturen [MqlBookInfo](#), das DOM Aufzeichnungen für das gewählte Symbol enthält.

```
bool MarketBookGet (
    string          symbol,      // Symbol
    MqlBookInfo&    book[]       // Referenz auf das Feld
);
```

### Parameter

*symbol*

[in] Symbolname.

*book[]*

[out] Verweis auf das Feld der DOM Aufzeichnungen. Feld kann vorläufig verteilt werden für ausreichende Zahl der Aufzeichnungen. Wenn das [dynamische Feld](#) nicht vorläufig im Operativspeicher verteilt wurde, wird das Kundenterminal dieses Feld selbstständig verteilen.

### Rückgabewert

Gibt true im Erfolgsfall zurück, anderenfalls false.

### Bemerkung

DOM muss vorläufig durch die Funktion [MarketBookAdd\(\)](#) geöffnet werden.

### Beispiel:

```
MqlBookInfo priceArray[];
bool getBook=MarketBookGet(NULL,priceArray);
if(getBook)
{
    int size=ArraySize(priceArray);
    Print("MarketBookInfo für",Symbol());
    for(int i=0;i<size;i++)
    {
        Print(i+": ",priceArray[i].price
            + "    Volume = "+priceArray[i].volume,
            " type = ",priceArray[i].type);
    }
}
else
{
    Print("Kann nicht Inhalt des Symbols DOM erhalten ",Symbol());
}
```

### Sehen Sie auch

[DOM Struktur](#), [Strukturen und Klassen](#)

## Zugang zu Zeitreihen und Daten der Indikatoren

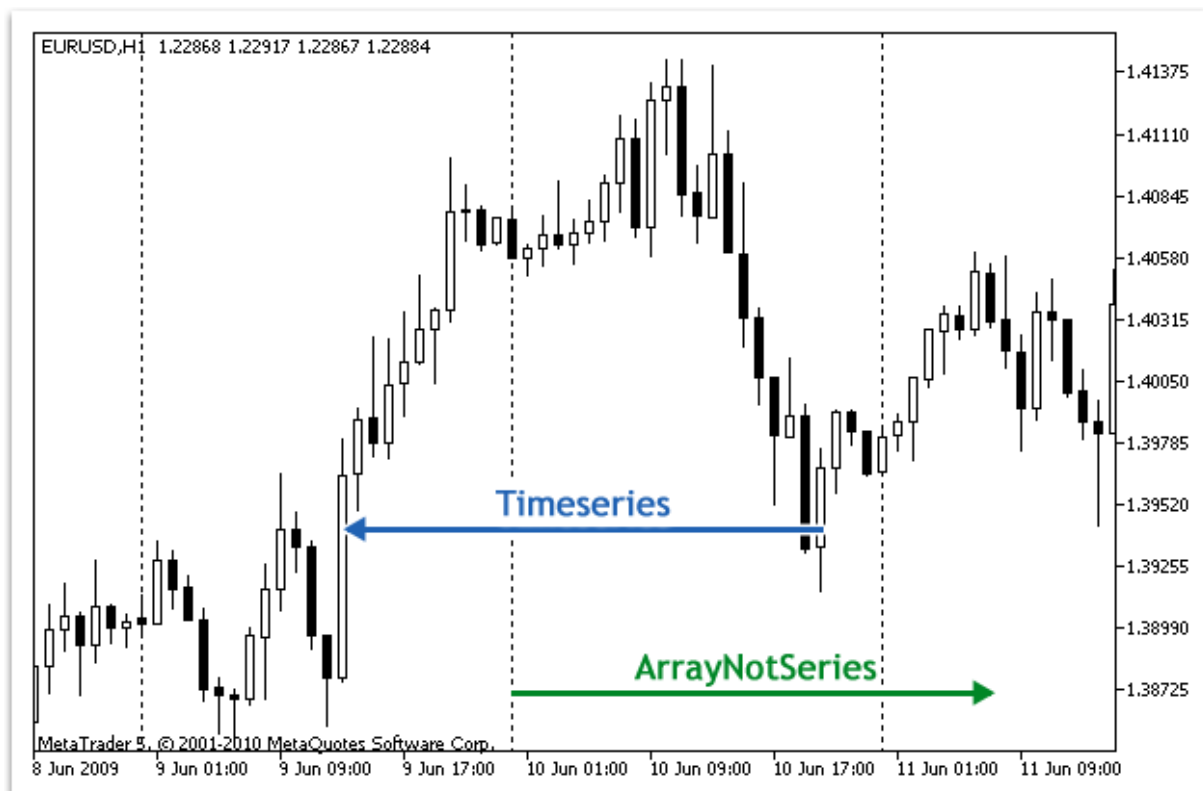
Funktionen für Arbeit mit Zeitreihen und Indikatoren. Zeitreihe unterscheidet sich vom Feld dadurch, dass Indizieren der Elemente einer Zeitreihe vom Ende des Feldes zum Anfang des Feldes durchgeführt wird (von den neuesten Daten zu den ältesten). Für Kopieren der Werte der Zeitreihe ist es empfehlenswert nur [dynamische Felder](#) zu verwenden, denn die Kopierfunktionen verteilen die erforderliche Größe der Felder, die die Werte empfangen.

Es gibt eine **wichtige Ausnahme** von dieser Regel: wenn Kopieren der Zeitreihen und Indikatorwerte oft durchgeführt werden muss, ZB bei jedem Aufruf [OnTick\(\)](#) in Experten oder bei jedem Aufruf [OnCalculate\(\)](#) in Indikatoren, ist es besser [statisch verteilte Felder](#) zu verwenden, denn **Operationen der Speicherverteilung** für dynamische Felder erfordert **zusätzliche Zeit** und das wird beim Testen oder Optimierung der Experte bedeutend sein.

Bei der Verwendung der Funktionen des Zugangs zu Zeitreihen und Werten des Indikators muss man Richtung des Indizierens beachten, das wird ausführlich im Abschnitt [Richtung des Indizierens in Feldern und Zeitreihen](#) beschrieben.

Zugang zu Werten der Indikatoren und Zeitreihen wird unabhängig davon durchgeführt, ob die angeforderten Daten fertig sind (der sogenannte [asynchroner Zugang](#)). Das ist kritisch bedeutsam für Berechnung der Benutzerindikatoren, darum geben die Funktionen des Typs *Copy...()* beim Fehlen der angeforderten Daten sofort den Fehler zurück. Aber beim Zugang von Experten und Scripts versucht man mehrmals Daten zu erhalten und dabei mit einer Pause, die die Zeit für Ladung fehlender Zeitreihen oder für Berechnung der Indikatorwerte bereitzustellen ist.

Im Abschnitt [Datenzugang organisieren](#) beschreibt man die Erhaltung, Aufbewahren und Anforderung der Preisdaten im Kundenterminal MetaTrader 5.



Es ist historisch bedingt, dass Zugang zu Daten im Preisfeld vom Ende der Daten durchgeführt wurde.

Physikalisch werden neue Daten an Ende des Feldes geschrieben, aber Index dieses Feldes ist immer 0. Index 0 im Feld-Zeitreihe bedeutet Daten der laufenden Bar, d. h. der Bar, die dem unbeeendeten Zeitintervall auf dem Timeframe entspricht.

Timeframe ist Zeitperiode, während der die einzelne Preisbar gebildet wird; insgesamt werden 21 [Standardtimeframes](#) vorbestimmt.

| Funktion                            | Massnahme                                                                                                                            |
|-------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">SeriesInfoInteger</a>   | Gibt die Information über den Zustand der historischen Daten zurück                                                                  |
| <a href="#">Bars</a>                | Gibt die die entsprechende PeriodeAnzahl der Bars in der Geschichte für das entsprechende Symbol und die entsprechende Periode       |
| <a href="#">BarsCalculated</a>      | Gibt die Anzahl der berechneten Daten im Anzeigepuffer oder -1 beim Fehler (Daten sind noch nicht berechnet)                         |
| <a href="#">IndicatorCreate</a>     | Gibt handle des angegebenen technischen Anzeigers zurück, das vom Feld des Typs Parameter <a href="#">MqlParam</a> erzeugt wurde     |
| <a href="#">IndicatorParameters</a> | Basierend auf das angegebenen Handle gibt Anzahl von Eingabeparametern des Indikators, sowie die Werte und Type der Parameter zurück |
| <a href="#">IndicatorRelease</a>    | Entfernt handle des Anzeigers und befreit den Berechnungsteil des Anzeigers, wenn niemand den verwendet                              |
| <a href="#">CopyBuffer</a>          | Bekommt Daten des angegebenen Puffers vom angegebenen Anzeiger im Feld                                                               |
| <a href="#">CopyRates</a>           | Bekommt historische Daten der Struktur <a href="#">Rates</a> für das angegebene Symbol und Periode                                   |
| <a href="#">CopyTime</a>            | Bekommt im Feld historische Daten über die Eröffnungszeit der Bars für das angegebene Symbol und Periode                             |
| <a href="#">CopyOpen</a>            | Bekommt im Feld historische Daten über die Eröffnungszeit der Bars für das angegebene Symbol und Periode                             |
| <a href="#">CopyHigh</a>            | Bekommt im Feld historische Daten über den maximalen Preis der Bars für das angegebene Symbol und Periode                            |
| <a href="#">CopyLow</a>             | Bekommt im Feld historische Daten über den minimalen Preis der Bars für das angegebene Symbol und Periode                            |
| <a href="#">CopyClose</a>           | Bekommt im Feld historische Daten über den Schlusspreis der Bars für das angegebene                                                  |

|                       | Symbol und Periode                                                                                   |
|-----------------------|------------------------------------------------------------------------------------------------------|
| <u>CopyTickVolume</u> | Bekommt im Feld historische Daten über Tickvolumen der Bars für das angegebene Symbol und Periode    |
| <u>CopyRealVolume</u> | Bekommt im Feld historische Daten über Handelsvolumen der Bars für das angegebene Symbol und Periode |
| <u>CopySpread</u>     | Bekommt im Feld historische Daten über Spreads für das angegebene Symbol und Periode                 |
| <u>CopyTicks</u>      | Erhält für die aktuelle Arbeitssitzung angesammelten Ticks ins Array ticks_array                     |

Abgesehen davon, dass durch die Funktion [ArraySetAsSeries\(\)](#) der Zugang zu Elementen in [Feldern](#) wie in Zeitreihen vorgegeben werden kann, muss man sich daran erinnern, dass Elemente des Feldes in derselben Reihenfolge aufbewahren werden, verändert wird nur Richtung des Indizierens. Diese Tatsache kann am folgenden Beispiel demonstriert werden:

```

datetime TimeAsSeries[];
//--- stellen wir Zugang zum Feld wie zur Timeserie ein
ArraySetAsSeries(TimeAsSeries,true);
ResetLastError();
int copied=CopyTime(NULL,0,0,10,TimeAsSeries);
if(copied<=0)
{
    Print("Nicht gelungen, die Eröffnungszeit für die letzten 10 Bars zu kopieren");
    return;
};
Print("TimeCurrent = ",TimeCurrent());
Print("ArraySize(Time) = ",ArraySize(TimeAsSeries));
int size=ArraySize(TimeAsSeries);
for(int i=0;i<size;i++)
{
    Print("TimeAsSeries["+i+"] = ",TimeAsSeries[i]);
};

datetime ArrayNotSeries[];
ArraySetAsSeries(ArrayNotSeries,false);
ResetLastError();
copied=CopyTime(NULL,0,0,10,ArrayNotSeries);
if(copied<=0)
{
    Print("Nicht gelungen, die Eröffnungszeit für die letzten 10 Bars zu kopieren");
    return;
};
size=ArraySize(ArrayNotSeries);
for(int i=size-1;i>=0;i--)
```

```
{  
    Print("ArrayNotSeries["+i+"] = ",ArrayNotSeries[i]);  
};
```

Im Ergebnis haben wir die Ausgabe, wie diese:

```
TimeCurrent = 2009.06.11 14:16:23  
ArraySize(Time) = 10  
TimeAsSeries[0] = 2009.06.11 14:00:00  
TimeAsSeries[1] = 2009.06.11 13:00:00  
TimeAsSeries[2] = 2009.06.11 12:00:00  
TimeAsSeries[3] = 2009.06.11 11:00:00  
TimeAsSeries[4] = 2009.06.11 10:00:00  
TimeAsSeries[5] = 2009.06.11 09:00:00  
TimeAsSeries[6] = 2009.06.11 08:00:00  
TimeAsSeries[7] = 2009.06.11 07:00:00  
TimeAsSeries[8] = 2009.06.11 06:00:00  
TimeAsSeries[9] = 2009.06.11 05:00:00  
  
ArrayNotSeries[9] = 2009.06.11 14:00:00  
ArrayNotSeries[8] = 2009.06.11 13:00:00  
ArrayNotSeries[7] = 2009.06.11 12:00:00  
ArrayNotSeries[6] = 2009.06.11 11:00:00  
ArrayNotSeries[5] = 2009.06.11 10:00:00  
ArrayNotSeries[4] = 2009.06.11 09:00:00  
ArrayNotSeries[3] = 2009.06.11 08:00:00  
ArrayNotSeries[2] = 2009.06.11 07:00:00  
ArrayNotSeries[1] = 2009.06.11 06:00:00  
ArrayNotSeries[0] = 2009.06.11 05:00:00
```

Wie es aus der Ausgabe ersichtlich ist, vermindert sich der Wert der Zeit für das Feld TimeAsSeries, d.h. wir bewegen uns von der Gegenwart zur Zukunft. Für das normale Feld ArrayNotSeries ist es umgekehrt - mit dem Anstieg des Indexes bewegen wir uns von der Vergangenheit zur Gegenwart.

Sehen Sie auch

[ArrayIsDynamic](#), [ArrayGetAsSeries](#), [ArraySetAsSeries](#), [ArrayIsSeries](#)



## Richtung des Indizierens in Feldern, Puffern und Zeitreihen

Alle Felder und Indikatorpuffer haben als Vorgabe die Richtung des Indizierens von links nach rechts. Index des ersten Elementes ist immer der Null gleich. So befindet sich das erste Element eines Feldes oder eines Indikatorpuffers mit Index 0 als Vorgabe an der linken Grenzstelle, das letzte Element befindet sich an der rechten Grenzstelle.

Indikatorpuffer ist ein [dynamisches Feld](#) des Typs double, dessen Größe vom Kundenterminal verwaltet wird, damit er der Anzahl von Bars entspricht, die für Berechnung des Indikators verwendet werden. Das normale dynamische Feld des Typs double wird als Indikatorpuffer durch die Funktion [SetIndexBuffer\(\)](#) vorgegeben. Für Indikatorpuffer braucht man nicht die Größe durch die Funktion [ArrayResize\(\)](#) vorgeben, das ausführende System des Terminals wird das selbstständig machen.

[Zeitreihen](#) sind Felder mit dem umgekehrten Indizieren, d.h. das erste Element einer Zeitreihe befindet sich an der linken Grenzstelle, das letzte Element befindet sich an der rechten Grenzstelle. Da Zeitreihen dafür bestimmt sind, historische Preisdaten für Finanzinstrumente aufzubewahren und unbedingt zeitbezogene Information enthalten, kann man sagen, dass sich die neuesten Daten in der Zeitreihe an der rechten Grenzstelle und die ältesten an der linken Grenzstelle befinden.

Darum hat Element mit Index 0 in der Zeitreihe Information über die letzte Quotation für Instrument. Wenn eine Zeitreihe Daten für für taegliches Timeframe darstellt, werden in der Null-Position Daten des laufenden nicht beendeten Tages geben, und in der Position mit Index 1 werden Daten von gestern aufbewahren.

## Veränderung der Richtung des Indizierens

Funktion [ArraySetAsSeries\(\)](#) erlaubt Zugangsverfahren zu Elementen eines dynamischen Feldes zu verändern, dabei wird die Ordnung der Datenaufbewahrung im Computerspeicher physisch nicht verändert. Die Funktion verändert Adressierungsart zu Feldelementen, darum wird der Inhalt des Feld-Rezipienten beim Kopieren eines Feldes in ein anderes Feld durch die Funktion [ArrayCopy\(\)](#) nicht von der Richtung des Indizierens im Feld-Quelle abhängen.

Man darf nicht Richtung des Indizierens für statisch verteilte Felder verändern. Auch wenn ein Feld als Parameter in die Funktion übertragen wurde, werden die Versuche, Indizieren innerhalb dieser Funktion zu verändern, zum nichts führen.

für Indikatorpuffer ist es auch wie für normale Felder moeglich, Richtung des Indizierens umgekehrt wie in einer Zeitreihe einzustellen, d.h. Zugang zur Null-Position in Indikatorpuffer wird Zugang zum letzten Wert im entsprechenden Indikatorpuffer bedeuten und das wird dem Indikatorwert in der allerletzten Bar entsprechen. Dabei wird physisch Ordnung der Daten im Indikatorpuffer nicht verändern, wie es schon erwaeht wurde.

## Erhalten von Preisdaten in Indikatoren

Jeder [Benutzerindikator](#) muss unbedingt die Funktion [OnCalculate\(\)](#) haben, der Preisdaten übertragen werden, die für Berechnung der Werte in Indikatorpuffern notwendig sind. Richtung des Indizierens in diesen übertragenen Feldern kann man durch die Funktion [ArrayGetAsSeries\(\)](#) erfahren .

[Die in die Funktion übertragenen](#) Felder stellen Preisdaten dar, d.h. diese Felder haben das Zeichen der Zeitreihe und die Funktion [ArrayIsSeries\(\)](#) gibt true bei der Pruefung dieser Felder zurück. Aber Richtung des Indizierens muss auf jeden Fall nur durch die Funktion [ArrayGetAsSeries\(\)](#) geprueft

werden.

Um nicht von den Default-Werten abhängig zu sein, muss man die Funktion [ArraySetAsSeries\(\)](#) für die Felder aufrufen, mit denen es zu arbeiten soll und die notwendige Richtung des Indizierens einstellen.

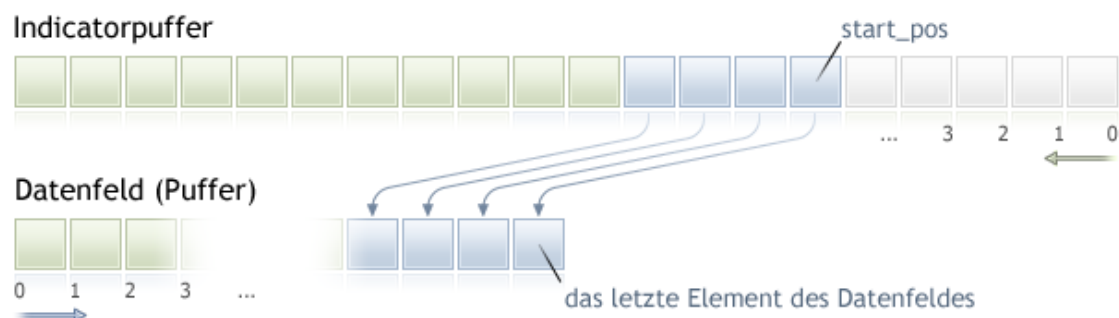
## Erhalten von Preisdaten und Indikatorwerten

In Experten, Indikatoren und Scripts haben alle Felder als Default die Richtung des Indizierens von links nach rechts. Wenn notwendig können die Werte der Zeitreihen für jedes Symbol und Timeframe in jedem mql5-Programm angefordert werden aber auch Indikatorwerte, die in jedem Symbol und Timeframe berechnet werden können.

für Erhaltung dieser Daten sind die Funktionen `Copy...()` bestimmt:

- [CopyBuffer](#) - Kopieren der Werte des Indikatorpuffers ins Feld des Typs double;
- [CopyRates](#) - Kopieren der Preisgeschichte ins Feld der Strukturen [MqlRates](#);
- [CopyTime](#) - Kopieren der Werte Time ins Feld des Typs datetime;
- [CopyOpen](#) - Kopieren der Werte Open ins Feld des Typs double;
- [CopyHigh](#) - Kopieren der Werte High ins Feld des Typs double;
- [CopyLow](#) - Kopieren der Werte Low ins Feld des Typs double;
- [CopyClose](#) - Kopieren der Werte Close ins Feld des Typs double;
- [CopyTickVolume](#) - Kopieren der Tickvolumen ins Feld des Typs long;
- [CopyRealVolume](#) - Kopieren der Boersenvolumen ins Feld des Typs long;
- [CopySpread](#) - Kopieren der Spreadgeschichte ins Feld des Typs int;

Alle diese Funktionen funktionieren gleich, darum reicht es, Mechanismus der Datenerhaltung am Beispiel `CopyBuffer()` zu betrachten. Es wird angenommen, dass alle angeforderten Daten dieselbe Richtung des Indizierens haben, wie in der Zeitreihe, dabei wird angenommen, dass in der Position mit Index 0 (Null) Daten der unbeeendeten Bar aufbewahren werden. Für Erhaltung des Zuganges zu diesen Daten muss man das notwendige Volumen ins Feld-Rezipient kopieren, ZB ins Feld *buffer*.



Beim Kopieren ist es notwendig, Startposition im Ausgangsfeld anzugeben, von der Daten ins Feld-Rezipient kopiert werden. Im Erfolgsfall wird ins Feld-Rezipient die angegebene Anzahl der Elementen vom Ausgangsfeld kopiert, in diesem Fall vom Indikatorpuffer. Dabei wird Kopieren immer so durchgeführt, wie es auf dem Bild gezeigt, unabhängig davon, welche Richtung des Indizierens im Feld-Rezipient angegeben wird.

Wenn Preisdaten voraussichtlich in einer Schleife mit einer großen Anzahl von Iterationen behandelt werden, ist es ratsam, dass Sie die Tatsache der erzwungenen Beendigung des Programms mit der Funktion IsStopped() überprüfen:

```
int copied=CopyBuffer(ma_handle, // Handle von Indikator
                    0,           // Index der Indikator-Puffer
                    0,           // Startposition für das Kopieren
                    number,      // Anzahl der Werten für das Kopieren
                    Buffer        // Das Array, das die Werte erhält
                    );

if(copied<0) return;
int k=0;
while(k<copied && !IsStopped())
{
    //--- Erhalten den Wert für den Index k
    double value=Buffer[k];
    // ...
    // Arbeit mit dem Wert value
    k++;
};
```

#### Beispiel:

```
input int per=10; // Periode der Exponente
int ma_handle;    // handle des Indikators
//+-----+
//| Expert initialization function |
//+-----+
int OnInit()
{
    //---
    ma_handle=iMA(_Symbol,0,per,0,MODE_EMA,PRICE_CLOSE);
    //---
    return(INIT_SUCCEEDED);
};
//+-----+
//| Expert tick function |
//+-----+
void OnTick()
{
    //---
    double ema[10];
    int copied=CopyBuffer(ma_handle, // handle des Indikators
                        0,           // Index des Indikatorpuffers
                        0,           // Anfangsposition für Kopieren
                        10,          // Anzahl der Werte für Kopieren
                        ema          // Feld-Rezipient der Werte
                        );

    if(copied<0) return;
    // .... weiterer Code
```

```
; }
```

Sehen Sie auch

[Datenzugang organisieren](#)

## Datenzugriff organisieren

In diesem Abschnitt werden die Fragen behandelt, die mit Erfassung, Aufbewahren und Anforderung der Preisdaten ([Timeserien](#)) verbunden sind.

### Datenerfassung vom Handelsserver

Bevor Preisdaten im Terminal MetaTrader 5 zugänglich sind, müssen sie erfasst und verarbeitet werden. Für Datenerfassung ist der Anschluss an den Handelsserver MetaTrader 5 erforderlich. Daten werden vom Server erfasst auf Anfrage des Terminals in der Form der gepackten Blocks der Minutenbars.

Der Mechanismus der Serverreferenz hängt nicht davon ab, wie die Anfrage initiiert wurde - vom Benutzer bei der Navigation durch Chart oder vom Programm in der Sprache MQL5.

### Aufbewahren der Zwischendaten

Die vom Server erfassten Daten werden verpackt und im speziellen Zwischenformat HCC aufbewahren. Daten für jedes Symbol werden in dem gesonderten Verzeichnis Terminalverzeichnis\bases\Name\_Servers\history\Name\_Symbols geschrieben. Z.B. Daten für Symbol EURUSD vom Handelsserver MetaQuotes-Demo werden sich im VerzeichnisKatalog\_Terminals\bases\MetaQuotes-Demo\history\EURUSD\ befinden.

Daten werden in die Dateien mit der Endung .hcc geschrieben, jede Datei bewahrt Daten der Minutenbars für ein Jahr. Z.B. die Datei 2009.hcc hat im Verzeichnis EURUSD Minutenbars für Symbol EURUSD für das Jahr 2009. Diese Daten werden für Vorbereitung der Preisdaten für alle Timeframes verwendet und sind nicht für den direkten Zugang bestimmt.

### Erhaltung der Daten des notwendigen Timeframes aus den Zwischendaten

Dateien im Format HCC sind Quelle für Bauen der Preisdaten für angeforderte Timeframes im Format HC. Daten im Format HC sind Timeserien, die für den schnellen Zugang am meisten maximal sind. Sie werden nur auf Anfrage des Charts oder des mql5-Programms erzeugt, das Volumen ist nicht mehr als Parameterwert "Max bars in charts", und sie sind für weitere Verwendung in Dateien mit Endung hc aufbewahren.

für Ressourcenökonomie werden Daten des Timeframes gespeichert und aufbewahren im Operativspeicher wenn notwendig, beim langen Fehlen der Aufrufe werden sie aus dem Operativspeicher befreit und in der Datei gespeichert. Für jedes Timeframe werden Daten vorbereiten unabhängig davon, ob es schon fertige Daten für andere Timeframes gibt. Die Regeln des Zusammenstellens und Zugänglichkeit der Daten sind für alle Timeframes gleich. D.h. Abgesehen davon, dass die Einheit der aufbewahrenen Daten im HCC Format Minutenbar ist, bedeutet das Vorhandensein der Daten im HCC Format nicht das Vorhandensein und Zugänglichkeit der Daten des Timeframes M1 im HC Format in derselben Anzahl.

Erhalten neuer Daten vom Server ruft automatischen Update der verwendeten Preisdaten im HC Format für alle timeframes und Nachkalkulation aller Anzeiger, die die Daten explizit verwenden als Eingabedaten für Berechnung.

### Parameter "Max bars in chart"

Parameter "Max bars in charts" beschränkt die Anzahl der Bars im Format HC, die für Charts,

Anzeiger und mql5-Programme zugaenglich sind. Diese Einschraenkung gilt für Daten aller Timeframes und ist in erster Linie für Sparen der Ressourcen bestimmt.

Bei der Einstellung der grossen Werte dieses Parameters muss an sich daran erinnern, dass wenn die Geschichte der Preisdaten für kleine Timeframes tief ist, kann der Speicher für Aufbewahren der Zeitreihen und Puffer hunderte Megabytes sein und Einschraenkungen des Operativspeichers für Programm des Kundenterminals erreichen (2GB für 32-Bit MS Windows-Anwendungen).

Veränderung des Parameters "Max bars in charts" tritt in Kraft nach Wiederanlauf des Kundenterminals. Die Veränderung dieses Parameters selbst führt weder zum Anruf des Servers, um zusätzliche Daten zu bekommen, noch zur Bildung zusätzlicher Bars der Zeitreihen. Anforderung der zusätzlichen Preisdaten und Erneuerung der Zeitreihen unter Beachtung der neuen Einschraenkung erfolgen entweder beim Scrolling des Charts zum Bereich mit den fehlenden Daten oder bei der Anforderung der fehlenden Daten vom mql5-Programm.

Das Volumen der vom Server angeforderten Daten entspricht der erforderlichen Anzahl der Bars des vorgegebenen Timeframes unter Beachtung vom Parameter "Max bars in charts". Einschraenkung, vorgegeben vom Parameter ist nicht strikt, und in einigen Faellen kann die Anzahl der zugaenglichen Daten für Timeframe bisschen groesser sein, als der laufende Wert des Parameters.

## Zugaenglichkeit der Daten

Anwesenheit der Daten im HCC Format oder im fertigen für Verwendung HC Format bedeutet nicht, dass diese Daten für Darstellung auf dem Chart oder für Verwendung in mql5-Programmen vollstaendig zugaenglich sind.

Beim Zugang zu Preisdaten oder zu Indikatorwerten aus mql5-Programmen muss man behalten, dass ihre Zugaenglichkeit für bestimmten Zeitpunkt oder seit dem bestimmten Zeitpunkt nicht garantiert wird. Das ist damit verbunden, dass zu Zwecken der Ressourcenökonomie wird in MetaTrader 5 nicht die ganze Kopie der erforderlichen Daten für mql5-Programm aufbewahren, sondern es wird ein direkter Zugang zur Datenbank des Terminals gegeben.

Preisgeschichte für alle Timeframes wird aus allgemeinen Daten des HCC Formats gebildet und jedes Update der Daten vom Server führt zum Update der Daten für alle Timeframes und Umbeechnung der Indikatoren. Deswegen kann der Datenzugang zurückgewiesen werden auch wenn diese Daten vor einem Moment zugaenglich waren.

## Synchronisierung der Terminaldaten und der Serverdaten

Da ein mql5-Programm Daten für jedes Symbol und Timeframe aufrufen kann, ist es moeglich, dass Daten der erforderlichen Serie im Terminal noch nicht formiert sind oder Preisdaten mit dem Handelsserver noch nicht synchronisiert sind. In diesem Fall ist Wartezeit schwer vorherzusagen.

Algorithmen, die Wartezeit Zyklen verwenden sind keine beste Entscheidung. der einzige Ausschluss ist in diesem Fall Scripts, denn sie keine andere Algorithmenwahl haben wegen des Fehlens der Ereignisverarbeitung. Für Benutzerindikatoren sind solche Algorithmen, wie auch andere Wartezeitzyklen strikt nicht empfohlen, denn sie führen zur Beendigung der Berechnung aller Indikatoren und einer anderen Verarbeitung der Preisdaten für das Symbol.

für Experten und Benutzerindikatoren ist es besser [Ereignismodel](#) der Verarbeitung zu verwenden. Wenn es bei der Verarbeitung des Ereignisses OnTick() oder OnCalculate() unmöglich war, alle erforderlichen Daten der Angeforderten Zeitreihe zu bekommen, muss man Ereignisbearbeiter

verlassen und vertrauen auf Datenzugang beim naechsten Aufruf des Bearbeiters.

## Beispiel eines Scripts für Laden der Geschichte

Betrachten wir ein Beispiel des Scripts, das Anforderung erfuehlt, Geschichte vom Handelsserver für das angegebene Symbol zu erhalten. Script ist dafür bestimmt, das erforderliche Instrument im Chart ablaufen zu lassen, Timeframe ist nicht von Bedeutung, denn, es wurde schon erwaeht, alle Preisdaten kommen vom Handelsserver als verpackte Minutendaten, aus denen danach jede vorbestimmte Zeitreihe gebildet wird.

Schreiben wir alle Handlungen bezüglich Datenerhaltung als eine gesonderte Funktion `CheckLoadHistory(symbol, timeframe, start_date)`:

```
int CheckLoadHistory(string symbol,ENUM_TIMEFRAMES period,datetime start_date)
{
;}
```

Funktion ist `CheckLoadHistory()` ist als universelle Funktion gedacht, die aus jedem Programm (Expert, Script oder Indikator) aufgerufen werden kann und darum muss sie drei Eingabeparameter haben: Symbolname, Periode und Anfangsdatum, von dem wir Preisgeschichte brauchen.

Geben wir in den Code der Funktion alle notwendigen Pruefungen ein, bevor wir die fehlende Geschichte fordern. Erstens, überzeugen wir uns, dass Symbolname und Periodenwert korrekt sind:

```
if(symbol==NULL || symbol=="") symbol=Symbol();
if(period==PERIOD_CURRENT) period=Period();
```

Dann überzeugen wir uns davon, dass das angegebene Symbol im Fenster MarketWatch zugaenglich ist, d.h. die Geschichte des angegebenen Symbols auch beim Senden der Anforderung zum Handelsserver zugaenglich ist. Wenn solches Symbol in MarketWatch fehlt fuegen wir das zu durch die Funktion [SymbolSelect\(\)](#).

```
if(!SymbolInfoInteger(symbol,SYMBOL_SELECT))
{
    if(GetLastError()==ERR_MARKET_UNKNOWN_SYMBOL) return(-1);
    SymbolSelect(symbol,true);
}
```

Jetzt ist es notwendig, Anfangsdatum der zugaenglichen Geschichte für das angegebene Paar Symbol/ Periode zu erhalten. Der Wert des Eingabeparameters `startdate`, der der Funktion `CheckLoadHistory()` übertragen wurde, kann in Intervall der schon zugaenglichen Geschichte gelangen und dann ist keine Anforderung zum Handelsserver erforderlich. Für Erhaltung des ersten Datums für Symbol-Periode ist zum jetzigen Zeitpunkt die Funktion [SeriesInfoInteger\(\)](#) mit dem Modifikator [SERIES\\_FIRSTDATE](#) bestimmt.

```
SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date);
if(first_date>0 && first_date<=start_date) return(1);
```

Die naechste Pruefung ist die Pruefung des Programmtyps, aus dem die Funktion aufgerufen wird. Erinnern wir daran, dass Senden der Anforderung, Zeitreihe mit derselben Periode wie beim Indikator zu update ist gar nicht erwuensenswert. Unerwuenschtheit der Datenanforderung für das selbe Symbol-Periode, wie beim Indikator ist dadurch bedingt, dass Update der historischen Daten in demselben Thread erfolgt, wo Indikator arbeitet. Darum ist clinch hoechstswarscheinlich. Für die Pruefung verwenden wir die Funktion [MQL5InfoInteger\(\)](#) mit dem Modifikator [MQL5\\_PROGRAM\\_TYPE](#).

```
if(MQL5InfoInteger(MQL5_PROGRAM_TYPE)==PROGRAM_INDICATOR && Period()==period && Symbol()<0)
    return(-4);
```

Wenn wir alle Prüfungen erfolgreich durchgelaufen haben, machen wir den letzten Versuch ohne Aufruf des Handelsservers. Zuerst erfahren wir über das Anfangsdatum, für das Minutendaten in HCC Format zugänglich sind. Rufen wir diesen Wert durch die Funktion `SeriesInfoInteger()` mit dem Modifikator `SERIES_TERMINAL_FIRSTDATE` auf und vergleichen ihn mit dem Parameterwert `start_date`.

```
if(SeriesInfoInteger(symbol, PERIOD_M1, SERIES_TERMINAL_FIRSTDATE, first_date))
{
    //--- there is loaded data to build timeseries
    if(first_date>0)
    {
        //--- force timeseries build
        CopyTime(symbol, period, first_date+PeriodSeconds(period), 1, times);
        //--- check date
        if(SeriesInfoInteger(symbol, period, SERIES_FIRSTDATE, first_date))
            if(first_date>0 && first_date<=start_date) return(2);
    }
}
```

Wenn nach aller Prüfung Durchführungsthread im Körper der Funktion `CheckLoadHistory()` noch bleibt, dann bedeutet es, es ist notwendig, die fehlenden Preisdaten vom Handelsserver anzufordern. Vor allem erfahren wir über den Wert "Max bars in chart" durch die Funktion `TerminalInfoInteger()`:

```
int maxBars=TerminalInfoInteger(TERMINAL_MAXBARS);
```

Diese Funktion brauchen wir, um Anforderung von extra Daten vorzubeugen. Dann klären wir das erste Datum in der Geschichte des Symbol auf dem Handelsserver auf (unabhängig von der Periode) durch die schon bekannte Funktion `SeriesInfoInteger()` mit dem Modifikator `SERIES_SERVER_FIRSTDATE`.

```
datetime first_server_date=0;
while(!SeriesInfoInteger(symbol, PERIOD_M1, SERIES_SERVER_FIRSTDATE, first_server_date))
    Sleep(5);
```

Da eine Anforderung eine asynchrone Operation ist, wird die Funktion mit der kleinen Verzögerung 5 Millisekunden aufgerufen bis die Variable `first_server_date` den Wert erhält oder Zyklusdurchführung vom Benutzer unterbrochen wird (`IsStopped()`, in diesem Fall gibt den Wert `true` zurück). Geben wir den korrekten Wert des Anfangsdatums an, von dem wir Preisdaten vom Handelsserver anfordern.

```
if(first_server_date>start_date) start_date=first_server_date;
if(first_date>0 && first_date<first_server_date)
    Print("Warning: first server date ", first_server_date,
          " for ", symbol, " does not match to first series date ", first_date);
```

Wenn das Anfangsdatum `first_server_date` auf dem Server niedriger als Anfangsdatum `first_date` für Symbol in HCC Format ist, wird im Journal die entsprechende Nachricht ausgegeben werden.

Jetzt sind wir bereit, Anforderung beim Handelsserver zu machen, um fehlende Preisdaten zu bekommen. Anforderung machen wir in Form eines Zyklus und fangen wir an, ihren Körper auszufüllen:



```

while(!IsStopped())
{
    //1. warten auf Synchronisierung zwischen umgebildeten Zeitreihe und Zwischenges
    //2. laufende Anzahl der Bars in dieser Zeitreihe erhalten
    // Wenn bars groesser ist, als MaxBarsInChart, können wir verlassen, die An
    //3. Erhalten wir Anfangsdatum first_date in der umgebildeten Zeitreihe und verg
    // start_date wenn first_date kleiner ist als start_date, können wir verlassen
    //4. Fordern wir vom Handelsserver einen neuen Teil der Geschichte - 100 Bars vo
    // zugaenglichen Bar mit Nr bars
}

```

Die ersten drei Punkte werden durch schon bekannte Methoden realisiert.

```

while(!IsStopped())
{
    //--- 1.Warten darauf, bis Umbildung der Zeitreihe beendet wird
    while(!SeriesInfoInteger(symbol,period,SERIES_SYNCHRONIZED) && !IsStopped())
        Sleep(5);
    //--- 2.fordern wir an, wieviel Bars wir jetzt haben
    int bars=Bars(symbol,period);
    if(bars>0)
    {
        //--- Anzahl der Bars ist mehr, als man auf dem Chart zeigen kann, verlassen
        if(bars>=maxBars) return(-2);
        //--- 3. erfahren wir über das Anfangsdatum in der Zeitreihe
        if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
            // Anfangsdatum ist früher als angefordert, Aufgabe ist erfuehlt
            if(first_date>0 && first_date<=start_date) return(0);
    }
    //4. Fordern wir vom Handelsserver einen neuen Teil der Geschichte von
    // der letzten zugaenglichen Bar mit der Nummer bars an - 100 Bars
}

```

Es bleibt der letzter vierte Punkt - unmittelbare Anforderung der Geschichte. Wir können nicht direkt Serveraufrufen, aber jede [Copy-Funktion](#) beim Fehlen der Geschichte in HCC Format initiiert automatisch das Senden dieser Anforderung vom Terminal zum Handelsserver. Da die Zeit des ersten Anfangsdatums in der Variable *first\_date* als das einfachste und natuerlichste Kriterium um Grad der Erfuellung der Anforderung einzuschuetzen, wird es am leichtesten sein, die Funktion [CopyTime\(\)](#) zu verwenden.

Beim Aufruf der Funktionen, durch die Kopieren jeder Daten von Zeitreihen erfolgt, muss man bemerken, dass Parameter *start* (Barnummer, mit der Kopieren der Preisdaten beginnt werden muss) immer innerhalb der zugaenglichen Geschichte des Terminlas sein muss. wenn wir nur 100 Bars haben, ist es sinnlos, 300 Bars zu kopieren versuchen, angefangen mit der Bar mit Index 500. Eine solche Anforderung wird als fehlerhafte angenommen werden und wird nicht verarbeitet, d.h. keine Geschichte vom Handelsserver wird geladen.

Eben darum werden wir 100 Bars kopieren, angefangen mit der Bar mit Index. Das garantiert sanfte Ladung der Geschichte vom Handelsserver, dabei wird mehr als angeforderte 100 Bars ausgeladen werden, Server gibt die Geschichte mit überschuss zurück.

```
int copied=CopyTime(symbol,period,bars,100,times);
```

Nach Kopieren muss man die Anzahl der kopiertenElemente analysieren, wenn der Versuch erfolglos geworden ist, wird der Wert der Variable *copied* gleich Null sein und der Wert des Counters *fail\_cnt* wird um 1 steigen. Nach 100 erfolglosen Versuche wird die Funktionsarbeit unterbrochen.

```

int fail_cnt=0;
...
int copied=CopyTime(symbol,period,bars,100,times);
if(copied>0)
{
    //--- pruefen wir die Daten
    if(times[0]<=start_date) return(0); // der kopierte Wert ist kleiner, fertig
    if(bars+copied>=max_bars) return(-2); // Anzahl der Bars ist mehr, als auf dem Chart
    // gezeigt werden kann, fertig

    fail_cnt=0;
}
else
{
    //--- nicht mehr als 100 erfolglose Versuche nacheinander
    fail_cnt++;
    if(fail_cnt>=100) return(-5);
    Sleep(10);
}

```

So, ist in der Funktion nicht nur korrekte Verarbeitung der laufenden Situation zu jedem Zeitpunkt der Durchführung organisiert, sondern wird auch Beendigungskode zurückgegeben, den wir nach Aufruf der Funktion CheckLoadHistory() für die Erhaltung der zusätzlichen Information verarbeiten können. ZB. auf dieser Weise:

```

int res=CheckLoadHistory(InpLoadedSymbol,InpLoadedPeriod,InpStartDate);
switch(res)
{
    case -1 : Print("Unbekanntes Symbol",InpLoadedSymbol);
    case -2 : Print("Angeforderte Anzahl der Bars ist mehr, als es auf Chart gezeigt");
    case -3 : Print("Durchführung wurde vom Benutzer unterbrochen");
    case -4 : Print("Indikator muss nicht eigene Namen laden");
    case -5 : Print("Laden ist nicht erfolgreich");
    case 0 : Print("alle Daten sind geladen");
    case 1 : Print("Anzahl der zugaenglichen Daten in der Zeitreihe ist genug");
    case 2 : Print("Zeitreihe ist aus vorhandenen Daten des Terminals gebildet");
    default : Print("Ergebnis der Durchführung ist nicht bestimmt");
}

```

Der vollstaendige Funktionscode ist in Script angegeben, das richtige Organisieren des Zuganges zu jeden Daten mit der Verarbeitung des Ergebnisses der Anforderung demonstriert.

**Kode:**

```

//+-----+
//|                                     TestLoadHistory.mq5 |
//|                                     Copyright 2009, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.02"
#property script_show_inputs
//--- input parameters
input string        InpLoadedSymbol="NZDUSD"; // Symbol to be load
input ENUM_TIMEFRAMES InpLoadedPeriod=PERIOD_H1; // Period to be load
input datetime      InpStartDate=D'2006.01.01'; // Start date
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    Print("Start load",InpLoadedSymbol+", "+GetPeriodName(InpLoadedPeriod), "from", InpSta
//---
    int res=CheckLoadHistory(InpLoadedSymbol,InpLoadedPeriod,InpStartDate);
    switch(res)
    {
        case -1 : Print("Unknown symbol ",InpLoadedSymbol); break;
        case -2 : Print("Requested bars more than max bars in chart"); break;
        case -3 : Print("Program was stopped"); break;
        case -4 : Print("Indicator shouldn't load its own data"); break;
        case -5 : Print("Load failed"); break;
        case 0 : Print("Loaded OK"); break;
        case 1 : Print("Loaded previously"); break;
        case 2 : Print("Loaded previously and built"); break;
        default : Print("Unknown result");
    }
//---
    datetime first_date;
    SeriesInfoInteger(InpLoadedSymbol,InpLoadedPeriod,SERIES_FIRSTDATE,first_date);
    int bars=Bars(InpLoadedSymbol,InpLoadedPeriod);
    Print("First date",first_date,"-",bars,"bars");
//---
}
//+-----+
//| |
//+-----+
int CheckLoadHistory(string symbol,ENUM_TIMEFRAMES period,datetime start_date)
{
    datetime first_date=0;
    datetime times[100];
//--- check symbol & period
    if(symbol==NULL || symbol=="") symbol=Symbol();
    if(period==PERIOD_CURRENT) period=Period();
//--- check if symbol is selected in the MarketWatch
    if(!SymbolInfoInteger(symbol,SYMBOL_SELECT))
    {
        if(GetLastError()==ERR_MARKET_UNKNOWN_SYMBOL) return(-1);
        SymbolSelect(symbol,true);
    }
//--- check if data is present
    SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date);
    if(first_date>0 && first_date<=start_date) return(1);
//--- don't ask for load of its own data if it is an indicator
    if(MQL5InfoInteger(MQL5_PROGRAM_TYPE)==PROGRAM_INDICATOR && Period()==period && Sym

```

```

        return(-4);
//--- second attempt
if(SeriesInfoInteger(symbol,PERIOD_M1,SERIES_TERMINAL_FIRSTDATE,first_date))
{
    //--- there is loaded data to build timeseries
    if(first_date>0)
    {
        //--- force timeseries build
        CopyTime(symbol,period,first_date+PeriodSeconds(period),1,times);
        //--- check date
        if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
            if(first_date>0 && first_date<=start_date) return(2);
    }
}
//--- max bars in chart from terminal options
int maxBars=TerminalInfoInteger(TERMINAL_MAXBARS);
//--- load symbol history info
datetime first_server_date=0;
while(!SeriesInfoInteger(symbol,PERIOD_M1,SERIES_SERVER_FIRSTDATE,first_server_date))
    Sleep(5);
//--- fix start date for loading
if(first_server_date>start_date) start_date=first_server_date;
if(first_date>0 && first_date<first_server_date)
    Print("Warning: first server date ",first_server_date,

```

```

        " for ",symbol," does not match to first series date ",first_date);
//--- load data step by step
int fail_cnt=0;
while(!IsStopped())
{
    //--- wait for timeseries build
    while(!SeriesInfoInteger(symbol,period,SERIES_SYNCHRONIZED) && !IsStopped())
        Sleep(5);
    //--- ask for built bars
    int bars=Bars(symbol,period);
    if(bars>0)
    {
        if(bars>=max_bars) return(-2);
        //--- ask for first date
        if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
            if(first_date>0 && first_date<=start_date) return(0);
    }
    //--- copying of next part forces data loading
    int copied=CopyTime(symbol,period,bars,100,times);
    if(copied>0)
    {
        //--- check for data
        if(times[0]<=start_date) return(0);
        if(bars+copied>=max_bars) return(-2);
        fail_cnt=0;
    }
    else
    {
        //--- no more than 100 failed attempts
        fail_cnt++;
        if(fail_cnt>=100) return(-5);
        Sleep(10);
    }
}
//--- stopped
return(-3);
}
//+-----+
//| gibt Zeilenwert der Periode zurück |
//+-----+
string GetPeriodName(ENUM_TIMEFRAMES period)
{
    if(period==PERIOD_CURRENT) period=Period();
    //---
    switch(period)
    {
        case PERIOD_M1: return("M1");
        case PERIOD_M2: return("M2");
        case PERIOD_M3: return("M3");
        case PERIOD_M4: return("M4");
        case PERIOD_M5: return("M5");
        case PERIOD_M6: return("M6");
        case PERIOD_M10: return("M10");
        case PERIOD_M12: return("M12");
        case PERIOD_M15: return("M15");
        case PERIOD_M20: return("M20");
        case PERIOD_M30: return("M30");
        case PERIOD_H1: return("H1");
        case PERIOD_H2: return("H2");
        case PERIOD_H3: return("H3");
        case PERIOD_H4: return("H4");
    }
}

```

```
case PERIOD_H6: return("H6");
case PERIOD_H8: return("H8");
case PERIOD_H12: return("H12");
case PERIOD_D1: return("Daily");
case PERIOD_W1: return("Weekly");
case PERIOD_MN1: return("Monthly");
}
//---
return("unknown period");
}
```

## SeriesInfoInteger

Gibt die Information über den Zustand der historischen Daten zurück. Es gibt 2 Varianten der Funktion.

Gibt den Wert der Eigenschaft direkt.

```
long SeriesInfoInteger(
    string          symbol_name,      // Symbolname
    ENUM_TIMEFRAMES timeframe,        // Periode
    ENUM_SERIES_INFO_INTEGER prop_id, // Identifikator der Eigenschaft
);
```

Gibt true oder false zurück abhängig von der erfolgreichen Durchführung der Funktion.

```
bool SeriesInfoInteger(
    string          symbol_name,      // Symbolname
    ENUM_TIMEFRAMES timeframe,        // Periode
    ENUM_SERIES_INFO_INTEGER prop_id, // Identifikator der Eigenschaft
    long&           long_var          // Variable für die Erhaltung der Info
);
```

### Parameter

*symbol\_name*

[in] Symbol.

*timeframe*

[in] Periode.

*prop\_id*

[in] Identifikator der angeforderten Eigenschaft, Aufzählungswert [ENUM\\_SERIES\\_INFO\\_INTEGER](#).

*long\_var*

[out] Variable, in die der Wert der angeforderten Eigenschaft gesetzt wird.

### Rückgabewert

Wert des Typs long für die erste Variante des Aufrufes.

für die zweite Variante des Aufrufs gibt true zurück, wenn diese Eigenschaft zugaenglich ist und der Wert in der Variable long\_var aufbewahren wird, anderenfalls gibt false zurück. Für die zusätzliche Information über den Fehler rufen Sie die Funktion GetLastError() auf.

### Beispiel:

```
void OnStart()
{
    //---
    Print("Anzahl der Bars für Symbol-Periode zur Zeit = ",
          SeriesInfoInteger(Symbol(), 0, SERIES_BARS_COUNT));

    Print("Das erste Datum für Symbol-Periode zur Zeit = ",
          (datetime)SeriesInfoInteger(Symbol(), 0, SERIES_FIRSTDATE));

    Print("Das erste Datum in der Geschichte für Symbol auf dem Server = ",
          (datetime)SeriesInfoInteger(Symbol(), 0, SERIES_SERVER_FIRSTDATE));

    Print("Symboldaten werden synchronisiert = ",
          (bool)SeriesInfoInteger(Symbol(), 0, SERIES_SYNCHRONIZED));
}
```



## Bars

Gibt die Anzahl der Bars in der Geschichte für das angegebene Symbol und Periode. Es gibt 2 Varianten der Funktion.

### Anzahl aller Bars in der Geschichte anfordern

```
int Bars(  
    string          symbol_name,      // Symbolname  
    ENUM_TIMEFRAMES timeframe,      // Periode  
);
```

### Anzahl der Bars für das gewählte Interval anfordern

```
int Bars(  
    string          symbol_name,      // Symbolname  
    ENUM_TIMEFRAMES timeframe,      // Periode  
    datetime        start_time,      // Anfangsdatum  
    datetime        stop_time       // Beendigungsdatum  
);
```

### Parameter

*symbol\_name*

[in] Symbol.

*timeframe*

[in] Periode.

*start\_time*

[in] Barzeit, die dem ersten Element entspricht.

*stop\_time*

[in] Barzeit, die dem letzten Element entspricht.

### Rückgabewert

Wenn Parameter *start\_time* und *stop\_time* angegeben werden, gibt die Funktion die Anzahl der Bars im Datenbereich. Wenn diese Parameter nicht angegeben werden, gibt die Funktion die gesamte Anzahl der Bars zurück.

### Bemerkung

Wenn Daten für eine Zeitreihe mit angegebenen Parametern beim Aufruf der Funktion `Bars()` im Terminal noch nicht formiert sind oder Daten einer Zeitreihe beim Funktionsaufruf mit dem Handelsserver nicht [synchronisiert](#) sind, gibt die Funktion den Null-Wert zurück.

### Beispiel:

```

int bars=Bars(_Symbol,_Period);
if(bars>0)
{
    Print("Anzahl der Bars in der Terminalgeschichte für Symbol-Periode zum jetzigen Zeitpunkt");
}
else //keine zugängliche Bars
{
    //--- Daten des Symbols sollen mit Daten auf dem Server synchronisiert werden
    bool synchronized=false;
    //--- Counter des Zyklus
    int attempts=0;
    // versuchen wir 5 mal, auf Synchronisierung zu warten
    while(attempts<5)
    {
        if(SeriesInfoInteger(Symbol(),0,SERIES_SYNCHRONIZED))
        {
            //--- Synchronisierung gemacht, beenden
            synchronized=true;
            break;
        }
        //--- steigen wir den Counter
        attempts++;
        //--- warten wir 10 Millisekunden auf die naechste Iteration
        Sleep(10);
    }
    //--- Zyklus beenden nach der Synchronisierung
    if(synchronized)
    {
        Print("Anzahl der Bars in der Terminalgeschichte für Symbol-Periode zum jetzigen Zeitpunkt");
        Print("Das erste Datum in der Terminalgeschichte für Symbol-Periode zum jetzigen Zeitpunkt: ",
              (datetime)SeriesInfoInteger(Symbol(),0,SERIES_FIRSTDATE));
        Print("Das erste Datum in der Geschichte für Symbol auf dem Server = ",
              (datetime)SeriesInfoInteger(Symbol(),0,SERIES_SERVER_FIRSTDATE));
    }
    //--- Synchronisierung der Daten nicht passiert
    else
    {
        Print("Misslungen Anzahl der Bars für zu bekommen ",_Symbol);
    }
}

```

Sehen Sie auch

[Ereignisbearbeiter](#)

## BarsCalculated

Gibt die Anzahl der berechneten Daten für den angeforderten Indikator.

```
int BarsCalculated(  
    int indicator_handle, // handle des Indikators  
);
```

### Parameter

*indicator\_handle*

[in] Handle des Indikators, das von der entsprechenden Indikatorfunktion erhalten wurde.

### Rückgabewert

Gibt die Anzahl der berechneten Daten im Indikatorpuffer oder -1 im Fall des Fehlers (Daten sind noch nicht berechnet).

### Bemerkung

Die Funktion ist in den Fällen sinnvoll, wenn die Daten des Indikators sofort nach seiner Erzeugung erhalten werden müssen (Handle des Indikators ist zugänglich).

### Beispiel:

```
void OnStart()  
{  
    double Ups[];  
    //--- stellen wir Zeichen der Zeitreihe für Felder  
    ArraySetAsSeries(Ups,true);  
    //--- erzeugen wir handle des Indikators Fractals  
    int FractalsHandle=iFractals(NULL,0);  
    //--- stellen wir Fehlerkode zurück  
    ResetLastError();  
    //--- versuchen wir, Werte des Indikators zu kopieren  
    int i,copied=CopyBuffer(FractalsHandle,0,0,1000,Ups);  
    if(copied<=0)  
    {  
        Sleep(50);  
        for(i=0;i<100;i++)  
        {  
            if(BarsCalculated(FractalsHandle)>0)  
                break;  
            Sleep(50);  
        };  
        copied=CopyBuffer(FractalsHandle,0,0,1000,Ups);  
        if(copied<=0)  
        {  
            Print("Mislungen, obere Fractals zu kopieren. Error = ",GetLastError(),  
                "i = ",i," copied = ",copied);  
            return;  
        };  
    }
```

```
    else
        Print("Gelungen, obere Fractals zu kopieren.",
            "i = ",i,"    copied = ",copied);
    ;}
else Print("Missslungen, obere Fractals zu kopieren. ArraySize = ",ArraySize(Ups));
;}
```

## IndicatorCreate

Gibt handle des angegebenen technischen Indikators, erzeugt auf der Grundlage des Feldes der Parameter des Typs [MqlParam](#).

```
int IndicatorCreate(
    string          symbol,           // Symbolname
    ENUM_TIMEFRAMES period,          // Periode
    ENUM_INDICATOR  indicator_type,   // Typ des Indikators aus der Z
    int             parameters_cnt=0,  // Anzahl der Parameter
    const MqlParam& parameters_array[]=NULL, // Feld der Parameter
);
```

### Parameter

*symbol*

[in] Symbolname auf deren Daten der Indikator berechnet werden wird. [NULL](#) bedeutet das laufende Symbol.

*period*

[in] Wert der Periode kann einer der Aufzählungswerte [ENUM\\_TIMEFRAMES](#) sein, 0 bedeutet das laufende Timeframe .

*indicator\_type*

[in] Typ des Indikators, kann einen der Aufzählungswerte [ENUM\\_INDICATOR](#) annehmen.

*parameters\_cnt*

[in] Anzahl der Parameter, die im Feld `parameters_array[]` übertragen werden. Elemente des Feldes haben den Spezialtyp der Struktur [MqlParam](#). Der Default-Wert ist Null - Parameter werden nicht übertragen. Wenn Nicht-Nullzahl der Parameter angegeben wird, ist der Parameter `parameters_array` obligatorisch. Man kann nicht mehr als 256 Parameter übertragen.

*parameters\_array[]=NULL*

[in] Feld des Typs `MqlParam`, dessen Elemente Typ und Wert jedes Eingabeparameters des [technischen Indikators](#) enthalten.

### Rückgabewert

Gibt handle des angegebenen technischen Indokators zurück, im Fall des Misserfolges gibt [INVALID\\_HANDLE](#) zurück.

### Bemerkung

Wenn handle des Indikators des Typs `IND_CUSTOM` erzeugt wird, muss das Feld *type* des ersten Elementes der Eingabeparameter `parameters_array` den Wert `TYPE_STRING` aus der Aufzählung [ENUM\\_DATATYPE](#) haben, und das Feld *string\_value* des ersten Elementes muss den Namen des Benutzerindikators enthalten. Benutzerindikator muss kompiliert werden (Datei mit Verbreitung EX5) und sich im Verzeichnis `MQL5/Indicators` des Kundenterminals oder eingebettetem Verzeichnis befinden.

Die für Testenerforderliche Indikatoren werden automatisch aus Aufruf der Funktionen `iCustom()` bestimmt, wenn der entsprechende Parameter von der [Konstantzeile](#) vorgegeben ist. Für andere Fälle (Verwendung der Funktion [IndicatorCreate\(\)](#) oder Verwendung einer nicht Konstantzeile im

Parameter, der den Namen des Indikator vorgibt) ist diese Eigenschaft `#property tester_indicator` erforderlich :

```
#property tester_indicator "indicator_name.ex5"
```

Wenn im Benutzerindikator die erste Form des Aufrufes verwendet wird, kann bei der Übertragung der Eingabeparameter vom letzten Parameter zusätzlich angegeben werden, welche Daten für seine Berechnung verwendet werden. Wenn Parameter "Apply to" nicht explizit angegeben wird, wird als Default die Berechnung für Werte PRICE\_CLOSE durchgeführt.

#### Beispiel:

```
void OnStart()
{
    MqlParam params[];
    int      h_MA, h_MACD;
    //--- create iMA("EURUSD", PERIOD_M15, 8, 0, MODE_EMA, PRICE_CLOSE);
    ArrayResize(params, 4);
    //--- set ma_period
    params[0].type      =TYPE_INT;
    params[0].integer_value=8;
    //--- set ma_shift
    params[1].type      =TYPE_INT;
    params[1].integer_value=0;
    //--- set ma_method
    params[2].type      =TYPE_INT;
    params[2].integer_value=MODE_EMA;
    //--- set applied_price
    params[3].type      =TYPE_INT;
    params[3].integer_value=PRICE_CLOSE;
    //--- create MA
    h_MA=IndicatorCreate("EURUSD", PERIOD_M15, IND_MA, 4, params);
    //--- create iMACD("EURUSD", PERIOD_M15, 12, 26, 9, h_MA);
    ArrayResize(params, 4);
    //--- set fast ma_period
    params[0].type      =TYPE_INT;
    params[0].integer_value=12;
    //--- set slow ma_period
    params[1].type      =TYPE_INT;
    params[1].integer_value=26;
    //--- set smooth period for difference
    params[2].type      =TYPE_INT;
    params[2].integer_value=9;
    //--- set indicator handle as applied_price
    params[3].type      =TYPE_INT;
    params[3].integer_value=h_MA;
    //--- create MACD based on moving average
    h_MACD=IndicatorCreate("EURUSD", PERIOD_M15, IND_MACD, 4, params);
    //--- use indicators
    //--- . . .
    //--- release indicators (first h_MACD)
    IndicatorRelease(h_MACD);
    IndicatorRelease(h_MA);
}
```

## IndicatorParameters

Basierend auf das angegebenen Handle gibt Anzahl von Eingabeparametern des Indikators, sowie die Werte und Type der Parameter zurück.

```
int IndicatorParameters(
    int indicator_handle,      // Handle des Indikators
    ENUM_INDICATOR& indicator_type, // Variable, um den Typ des Indikators zu erhalten
    MqlParam& parameters[]    // Array, um die Parameter zu erhalten
);
```

### Parameter

*indicator\_handle*

[in] Handle auf den Indikator, für den Sie die Anzahl der Parameter, auf die er berechnet wurde, erhalten möchten.

*indicator\_type*

[out] Variable vom Typ [ENUM\\_INDICATOR](#), in die der Typ des Indikators geschrieben wird.

*parameters[]*

[out] Ein dynamisches Array für Werte vom Typ [MqlParam](#), in das die Liste der Indikator-Parameter geschrieben wird. Die Funktion `IndicatorParameters()` gibt die Größe des Arrays zurück.

### Rückgabewert

Anzahl der Eingabeparameter des Indikators mit dem angegebenen Handle, im Falle eines Fehlers gibt -1 zurück. Um Informationen über den Fehler zu erhalten, rufen Sie die Funktion [GetLastError\(\)](#) an.

### Beispiel:

```
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    //--- Die Anzahl der Fenster auf dem Chart (es gibt immer mindestens ein Hauptfenster)
    int windows=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
    //--- Für alle Fenster des Charts machen
    for(int w=0;w<windows;w++)
    {
        //--- Die Anzahl der Indikatoren in diesem Fenster/Unterfenster
        int total=ChartIndicatorsTotal(0,w);
        //--- Für alle Indikatoren im Fenster machen
        for(int i=0;i<total;i++)
        {
            //--- Kurzname des Indikators erhalten
            string name=ChartIndicatorName(0,w,i);
            //--- Handle des Indikators erhalten
            int handle=ChartIndicatorGet(0,w,name);
```

```

//--- In Log schreiben
PrintFormat("Window=%d, indicator #d, handle=%d",w,i,handle);
//---
MqlParam parameters[];
ENUM_INDICATOR indicator_type;
int params=IndicatorParameters(handle,indicator_type,parameters);
//--- Nachrichtenkopf
string par_info="Short name "+name+", type "
               +EnumToString(ENUM_INDICATOR(indicator_type))+"\r\n";
//---
for(int p=0;p<params;p++)
{
    par_info+=StringFormat("parameter %d: type=%s, long_value=%d, double_value=%d, string_value=%s",
                           p,
                           EnumToString((ENUM_DATATYPE)parameters[p].type),
                           parameters[p].integer_value,
                           parameters[p].double_value,
                           parameters[p].string_value
                           );
}
Print(par_info);
}
//--- Getan für alle Indikatoren im Fenster
}
//---
}

```

Sehen Sie auch

[ChartIndicatorGet\(\)](#)



## IndicatorRelease

Entfernt handle des Indikators und setzt Berechnungsteil des Indikators frei, wenn es nicht von jemandem verwendet wird.

```
bool IndicatorRelease(  
    int indicator_handle // handle des Indikators  
);
```

### Rückgabewert

Gibt true im Erfolgsfall zurück, anderenfalls false.

### Bemerkung

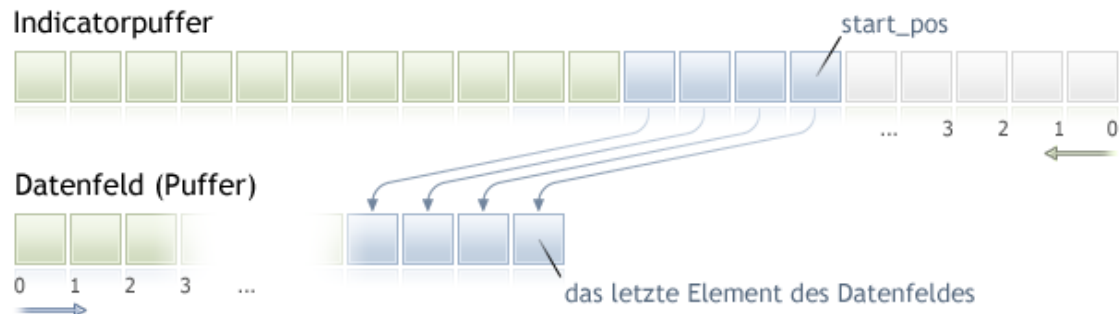
Funktion erlaubt handle des Indikators zu entfernen, wenn es nicht brauchbar ist und spart den Speicher damit. Handle wird sofort entfernt, Berechnungsteil des Indikators wird nach einiger Zeit durchgeführt (wenn er nicht mehr aufgerufen wird).

### Beispiel:

```
//+-----+
//|                                     Test_IndicatorRelease.mq5 |
//|                                     Copyright 2009, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- input parameters
input int          MA_Period=15;
input int          MA_shift=0;
input ENUM_MA_METHOD MA_smooth=MODE_SMA;
input ENUM_APPLIED_PRICE price=PRICE_CLOSE;
//--- wir werden handle des Indikators aufbewahren
int MA_handle;
//+-----+
//| Expert initialization function |
//+-----+
int OnInit()
{
//--- erzeugen wir handle des Indikators
MA_handle=iMA(Symbol(),0,MA_Period,MA_shift,MA_smooth,price);
//---
return(INIT_SUCCEEDED);
}
//+-----+
//| Expert tick function |
//+-----+
void OnTick()
{
//--- wenn der Wert der globalen Variable noch nicht festgelegt wird
if(GlobalVariableCheck("MA_value")==0)
{
//--- dynamisches Feld für Erhaltung der Werte des Indikators
double v[];
//--- erhalten wir Werte des Indikators in zwei letzten Bars
if(CopyBuffer(MA_handle,0,0,2,v)==2 && v[1]!=EMPTY_VALUE)
{
//--- speichern wir Wert der globalen Variable in der vorletzten Bar
if(GlobalVariableSet("MA_value",v[1]))
{
//--- befreien wir handle des Indikators
if(!IndicatorRelease(MA_handle))
Print("IndicatorRelease() failed. Error ",GetLastError());
}
}
}
//---
}
```

## CopyBuffer

Bekommt Daten des angegebenen Puffers in der angegebenen Zahl. Es gibt 3 Varianten der Funktion.



Abzählen der Elemente der kopierten Daten (Indicatorpuffer mit dem Index `buffer_num`) von der Startposition wird von der Gegenwart zur Vergangenheit durchgeführt, d. h. die Startposition, 0. bedeutet die laufende Bar (Indikatorwert für die laufende Bar).

Als Feld-Rezipient `buffer[]` ist es wünschenswert, ein [dynamisches Feld](#) zu verwenden, denn die Funktion `CopyBuffer()` versucht, die Größe des Feld-Rezipienten für Größe der kopierten Daten zu verteilen. Wenn als Feld-Rezipient `buffer[]` ein Indicatorpuffer auftritt (Feld, das für Aufbewahren der Indikatorwerte durch die Funktion [SetIndexBufer\(\)](#) verteilt wurde), ist partielles Kopieren zugelassen. Beispiel kann man im Benutzerindikator `Awesome_Oscillator.mq5` in der Standardlieferung des Terminals sehen.

Wenn es erforderlich ist, Indikatorwerte in ein anderes Feld (nicht Indicatorpuffer) partiell zu kopieren, muss man dazu ein Zwischenfeld verwenden, in das die erforderliche Anzahl kopiert wird. Und von diesen Zwischenfeld wird die erforderliche Anzahl der Werte in die notwendigen Stellen des Feld-Rezipienten elementenweise kopiert.

Wenn man die im voraus bekannte Anzahl der Daten kopieren muss, ist es besser dafür einen [statisch verteilten Puffer](#) zu verwenden, um unnötige Neuverteilung des Speichers zu vermeiden.

Es ist egal, welche Eigenschaft das Empfangsfeld hat - `as_series=true` oder `as_series=false`, Daten werden so kopiert, dass das älteste Element am Anfang des physischen Speichers sein wird, der für das Feld verteilt wurde. Es gibt 3 Varianten der Funktion.

### Aufruf nach Anfangsposition und Anzahl der angeforderten Elemente

```
int CopyBuffer(
    int      indicator_handle,    // handle des Indikators
    int      buffer_num,         // Nummer des Puffers des Indikators
    int      start_pos,          // Anfangsposition
    int      count,              // Anzahl für Kopieren
    double   buffer[])           // Feld, wohin Daten kopiert werden
);
```

### Aufruf nach Anfangsdatum und Anzahl der angeforderten Elemente

```
int CopyBuffer(
    int      indicator_handle,    // handle des Indikators
```

```

int      buffer_num,           // Nummer des Puffers des Indikators
datetime start_time,          // Anfangsdatum
int      count,               // Anzahl für Kopieren
double   buffer[]              // Feld, wohin Daten kopiert werden
);

```

#### Aufruf nach Anfangs- und Beendigungsdatum des angeforderten Zeitintervals

```

int CopyBuffer(
    int      indicator_handle,   // handle des Indikators
    int      buffer_num,        // Nummer des Puffers des Indikators
    datetime start_time,        // Anfangsdatum
    datetime stop_time,         // Beendigungsdatum
    double   buffer[]           // Feld, wohin Daten kopiert werden
);

```

#### Parameter

*indicator\_handle*

[in] Handle des Indikators, das durch die Indikatorfunktion zurückgegeben wird.

*buffer\_num*

[in] Nummer des Puffers des Indikators.

*start\_pos*

[in] Nummer des ersten kopierten Elementes.

*count*

[in] Anzahl der kopierten Elemente.

*start\_time*

[in] Barzeit, die dem ersten Element entspricht.

*stop\_time*

[in] Barzeit, die dem letzten Element entspricht.

*buffer[]*

[out] Feld des Typs [double](#).

#### Rückgabewert

Anzahl der kopierten Elemente des Feldes oder -1 beim [Fehler](#).

#### Bemerkung

Wenn die angeforderten Timeserien noch nicht gebildet sind oder müssen vom Server geladen werden, kehrt die Funktion sofort bei der Anforderung der Daten aus dem Indikator -1 zurück, aber dabei wird das Prozess der Ladung/Bildung initialisiert.

Bei der Anforderung der Daten aus dem Experten oder Script wird die [Ladung vom Server](#) initialisiert, wenn dieses Terminal keine lokale Daten hat oder Bildung der notwendigen Timeserie fängt an, wenn Daten aus der lokalen Geschichte gebildet werden kann, aber sie sind noch nicht fertig. Funktion kehrt diese Anzahl der Daten, die zum Augenblick des Timeoutsablaufs fertig sind.

## Beispiel:

```
//+-----+
//|                                     TestCopyBuffer3.mq5 |
//|                                     Copyright 2009, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+

#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots  1
//---- plot MA
#property indicator_label1  "MA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- input parameters
input bool          AsSeries=true;
input int           period=15;
input ENUM_MA_METHOD smootMode=MODE_EMA;
input ENUM_APPLIED_PRICE price=PRICE_CLOSE;
input int           shift=0;
//--- indicator buffers
double             MABuffer[];
int                ma_handle;
//+-----+
//| Custom indicator initialization function |
//+-----+

int OnInit()
{
//--- indicator buffers mapping
SetIndexBuffer(0,MABuffer,INDICATOR_DATA);
Print("Parameter AsSeries =",AsSeries);
Print("Indikatorpuffer nach SetIndexBuffer() ist eine Zeitreihe = ",
      ArrayGetAsSeries(MABuffer));
//--- set short indicator name
IndicatorSetString(INDICATOR_SHORTNAME,"MA("+period+")"+AsSeries);
//--- set AsSeries(depends from input parameter)
ArraySetAsSeries(MABuffer,AsSeries);
Print("Indikatorpuffer nach ArraySetAsSeries(MABuffer,true); ist eine Zeitreihe = ",
      ArrayGetAsSeries(MABuffer));
//---
ma_handle=iMA(Symbol(),0,period,shift,smootMode,price);
return(INIT_SUCCEEDED);
;}
```

```
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
    //--- check if all data calculated
    if(BarsCalculated(ma_handle)<rates_total) return(0);
    //--- we can copy not all data
    int to_copy;
    if(prev_calculated>rates_total || prev_calculated<=0) to_copy=rates_total;
    else
    {
        to_copy=rates_total-prev_calculated;
        //--- last value is always copied
        to_copy++;
    };
    //--- try to copy
    if(CopyBuffer(ma_handle,0,0,to_copy,MABuffer)<=0) return(0);
    //--- return value of prev_calculated for next call
    return(rates_total);
};
//+-----+
```

Im oben angeführten Beispiel ist die Ausfüllung des Indikatorpuffers durch Werte eines anderen Indikatorpuffers vom Indikator in demselben Symbol/Periode dargestellt.

Sehen Sie ein umfassenderes Beispiel von Anfrage der historischen Daten in [Methode der Objektbindung](#). Der Skript zeigt, wie Daten von Indikator [iFractals](#) an den letztem 1000 Balken zu erhalten und wie dann die letzten 10 Fraktale nach oben und 10 Fraktale nach unten auf dem Chart anzuzeigen. Sie können eine ähnliche Technik für alle Indikatoren, die fehlende Werte haben und typischerweise unter Verwendung der folgenden [Stil](#) hergestellt werden, verwenden:

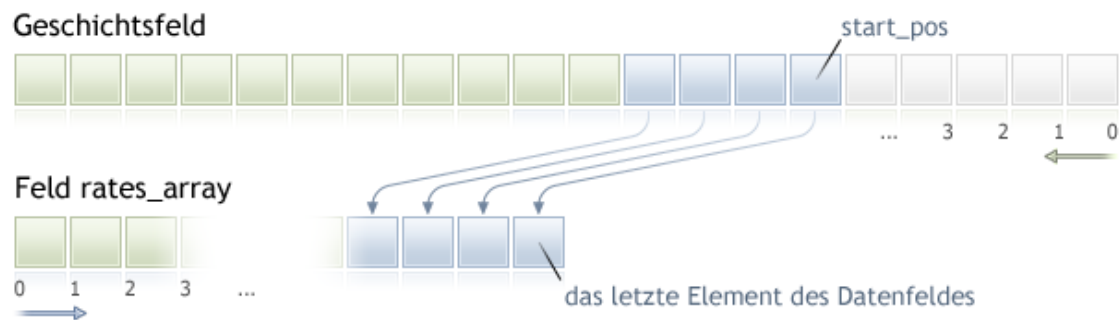
- [DRAW\\_SECTION](#),
- [DRAW\\_ARROW](#),
- [DRAW\\_ZIGZAG](#),
- [DRAW\\_COLOR\\_SECTION](#),
- [DRAW\\_COLOR\\_ARROW](#),
- [DRAW\\_COLOR\\_ZIGZAG](#).

Sehen Sie auch

[Eigenschaften der Benutzerindikatoren](#), [SetIndexBuffer](#)

## CopyRates

Bekommt im Feld `rates_array` historische Daten der Struktur [MqlRates](#) des angegebenen Symbol-Periode in der angegebenen Menge. Abzählen der Elemente von der Startposition wird von der Gegenwart zur Vergangenheit durchgeführt, d. h. Startposition, 0, bedeutet die laufende Bar.



Beim Kopieren der im voraus unbekannten Anzahl der Daten ist es empfehlenswert, als Feld-Rezipient ein dynamisches Feld zu verwenden, denn wenn die Anzahl der Daten weniger (oder mehr) ist als ein Feld enthalten kann, versucht man, ein Feld so zu verteilen, dass angeforderte Daten vollständig hineinpassen.

Wenn man die im voraus bekannte Anzahl der Daten kopieren muss, ist es besser dafür einen [statisch verteilten Puffer](#) zu verwenden, um unnötige Neuverteilung des Speichers zu vermeiden.

Es ist egal, welche Eigenschaft das Empfangsfeld hat - `as_series=true` oder `as_series=false`, Daten werden so kopiert, dass das älteste Element am Anfang des physischen Speichers sein wird, der für das Feld verteilt wurde. Es gibt 3 Varianten der Funktion.

### Aufruf nach Anfangsposition und Anzahl der angeforderten Elemente

```
int CopyRates(
    string          symbol_name,      // Symbolname
    ENUM_TIMEFRAMES timeframe,        // Periode
    int             start_pos,        // Anfangsposition
    int             count,            // Anzahl für Kopieren
    MqlRates        rates_array[]     // Feld, wohin Daten kopiert werden
);
```

### Aufruf nach Anfangsdatum und Anzahl der angeforderten Elemente

```
int CopyRates(
    string          symbol_name,      // Symbolname
    ENUM_TIMEFRAMES timeframe,        // Periode
    datetime        start_time,       // Anfangsdatum
    int             count,            // Anzahl für Kopieren
    MqlRates        rates_array[]     // Feld, wohin Daten kopiert werden
);
```

### Aufruf nach Anfangs- und Beendigungsdatum des angeforderten Zeitintervalls

```
int CopyRates(
```



```

string      symbol_name,      // Symbolname
ENUM_TIMEFRAMES timeframe,    // Periode
datetime    start_time,      // Anfangsdatum
datetime    stop_time,       // Beendigungsdatum
MqlRates    rates_array[]    // Feld, wohin Daten kopiert werden
);

```

### Parameter

*symbol\_name*

[in] symbol.

*timeframe*

[in] Periode.

*start\_time*

[in] Barzeit, die dem ersten Element entspricht.

*start\_pos*

[in] Nummer des ersten kopierten Elementes.

*count*

[in] Anzahl der kopierten Elemente.

*stop\_time*

[in] Barzeit, die dem letzten Element entspricht.

*rates\_array[]*

[out] Feld des Typs [MqlRates](#).

### Rückgabewert

Anzahl der kopierten Elemente oder -1 beim [Fehler](#).

### Bemerkung

Wenn das Interval der angeforderten Daten ausser den zugänglichen Daten auf dem Server ist, kehrt die Funktion -1 zurück. Falls die Daten ausserhalb [TERMINAL\\_MAXBARS](#) angefordert werden (maximale Anzahl der Bars auf dem Chart) gibt die Funktion auch -1 zurück.

Wenn die angeforderten Timeserien noch nicht gebildet sind oder müssen vom Server geladen werden, kehrt die Funktion sofort bei der Anforderung der Daten aus dem Indikator -1 zurück, aber dabei wird das Prozess der Ladung/Bildung initialisiert.

Bei der Anforderung der Daten aus dem Experten oder Script wird die [Ladung vom Server](#) initialisiert, wenn dieses Terminal keine lokale Daten hat oder Bildung der notwendigen Timeserie fängt an, wenn Daten aus der lokalen Geschichte gebildet werden kann, aber sie sind noch nicht fertig. Funktion kehrt diese Anzahl der Daten, die zum Augenblick des Timeoutsablaufs fertig sind, aber Ladung der Geschichte wird fortsetzen und bei der weiteren analogen Anforderung kehrt die Funktion schon mehr Daten zurück.

Bei der Anforderung der Daten nach Anfangsdatum und Anzahl der angeforderten Elemente kehren nur die Daten zurück, deren Datum kleiner (früher) oder gleich dem angegebenen Datum ist. Dabei wird das Interval mit Genauigkeit bis zur Sekunde vorgegeben und berücksichtigt. D.h. die Zeit der

Öffnung jeder Bar, für die der Wert zurückgegeben wird (Volumen, Spread, Wert im Indikatorpuffer, Preis Open, High, Low, Close oder Zeit der Öffnung Time), ist immer gleich oder kleiner als das angegebene Datum.

Bei der Anforderung der Daten im vorgegebenen Datenbereich kehren nur die Daten zurück, die in das angeforderte Interval kommen, dabei wird das Interval mit Genauigkeit bis zur Sekunde vorgegeben und berücksichtigt. D.h. die Zeit der Öffnung jeder Bar, für die der Wert zurückgegeben wird (Volumen, Spread, Wert im Indikatorpuffer, Preis Open, High, Low, Close oder Zeit der Öffnung Time) befindet sich immer im angeforderten Interval.

So wenn der laufende Wochentag Samstag ist, kehrt die Funktion beim Versuch, Daten in dem wöchentlichen Timeframe mit Andeutung *start\_time=Letzter\_Dienstag* und *stop\_time=Letzter\_Freitag* zu kopieren, 0 zurück, denn die Eröffnungszeit im wöchentlichen Timeframe ist immer der Sonntag, aber keine angegebene Bar gerät in den angegebenen Bereich.

Wenn man muss, den ersten Wert zu erhalten, der der laufenden nicht beendeten Bar entspricht, kann man die erste Aufrufform mit der Andeutung *start\_pos=0* und *count=1* verwenden.

#### Beispiel:

```
void OnStart()
{
//---
MqlRates rates[];
ArraySetAsSeries(rates,true);
int copied=CopyRates(Symbol(),0,0,100,rates);
if(copied>0)
{
Print("Anzahl der kopierten Bars: "+copied);
string format="open = %G, high = %G, low = %G, close = %G, volume = %d";
string out;
int size=fmin(copied,10);
for(int i=0;i<size;i++)
{
out=i+": "+TimeToString(rates[i].time);
out=out+" "+StringFormat(format,
                        rates[i].open,
                        rates[i].high,
                        rates[i].low,
                        rates[i].close,
                        rates[i].tick_volume);

Print(out);
};
};
else Print("Nicht gelungen, historische Daten nach Symbol zu erhalten ",Symbol());
};
```

Sehen Sie ein umfassenderes Beispiel von Anfrage der historischen Daten in [Methode der Objektbindung](#). Der Skript zeigt, wie Daten von Indikator [iFractals](#) an den letztem 1000 Balken zu erhalten und wie dann die letzten 10 Fraktale nach oben und 10 Fraktale nach unten auf dem Chart anzuzeigen. Sie können eine ähnliche Technik für alle Indikatoren, die fehlende Werte haben und

typischerweise unter Verwendung der folgenden [Stil](#) hergestellt werden, verwenden:

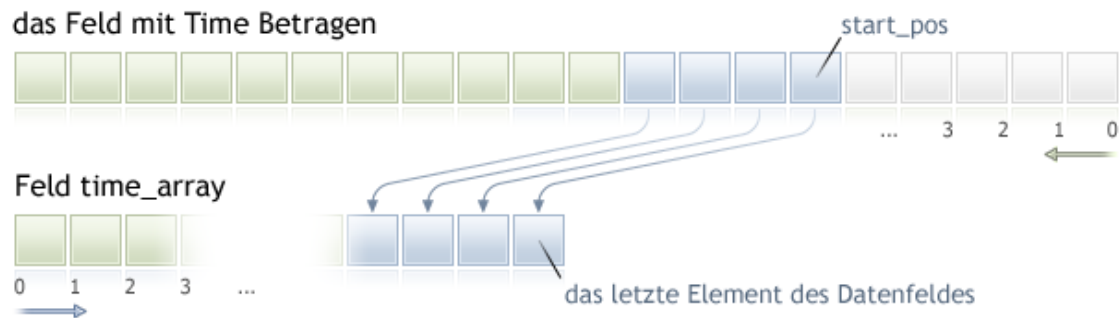
- [DRAW\\_SECTION](#),
- [DRAW\\_ARROW](#),
- [DRAW\\_ZIGZAG](#),
- [DRAW\\_COLOR\\_SECTION](#),
- [DRAW\\_COLOR\\_ARROW](#),
- [DRAW\\_COLOR\\_ZIGZAG](#).

Sehen Sie auch

[Strukturen und Klassen](#), [TimeToString](#), [StringFormat](#)

## CopyTime

Funktion erhält im Feld `time_array` historische Daten der Eröffnungszeit der Bars für das angegebene Paar Symbol-Periode in der angegebenen Menge. Es muss bemerkt werden, dass Abzählen der Elemente von der Startposition von der Gegenwart zur Vergangenheit durchgeführt wird, d.h. die Anfangsposition, 0, bedeutet die laufende Bar.



Beim Kopieren der im voraus unbekannten Anzahl der Daten ist es empfehlenswert, als Feld-Rezipient ein [dynamisches Feld](#) zu verwenden, denn wenn die Anzahl der Daten weniger (oder mehr) ist als ein Feld enthalten kann, versucht man, ein Feld so zu verteilen, dass angeforderte Daten vollständig hineinpassen.

Wenn man die im voraus bekannte Anzahl der Daten kopieren muss, ist es besser dafür einen [statisch verteilten Puffer](#) zu verwenden, um unnötige Neuverteilung des Speichers zu vermeiden. .

Es ist egal, welche Eigenschaft das Empfangsfeld hat - `as_series=true` oder `as_series=false`, Daten werden so kopiert, dass das älteste Element am Anfang des physischen Speichers sein wird, der für das Feld verteilt wurde. Es gibt 3 Varianten der Funktion.

### Aufruf nach Anfangsposition und Anzahl der angeforderten Elemente

```
int CopyTime(
    string          symbol_name,      // Symbolname
    ENUM_TIMEFRAMES timeframe,        // Periode
    int             start_pos,        // Anfangsposition
    int             count,            // Anzahl für Kopieren
    datetime        time_array[]      // Feld für Kopieren der Eröffnungszeit
);
```

### Aufruf nach Anfangsdatum und Anzahl der angeforderten Elemente

```
int CopyTime(
    string          symbol_name,      // Symbolname
    ENUM_TIMEFRAMES timeframe,        // Periode
    datetime        start_time,       // Anfangsdatum
    int             count,            // Anzahl für Kopieren
    datetime        time_array[]      // Feld für Kopieren der Eröffnungszeit
);
```

### Aufruf nach Anfangs- und Beendigungsdatum des angeforderten Zeitintervalls

```

int CopyTime (
    string          symbol_name,      // Symbolname
    ENUM_TIMEFRAMES timeframe,       // Periode
    datetime        start_time,      // Anfangsdatum
    datetime        stop_time,       // Beendigungsdatum
    datetime        time_array[]     // Feld für Kopieren der Eröffnungszeit
);

```

### Parameter

*symbol\_name*

[in] Symbol.

*timeframe*

[in] Periode.

*start\_pos*

[in] Nummer des ersten kopierten Elementes.

*count*

[in] Anzahl der kopierten Elemente.

*start\_time*

[in] Barzeit, die dem ersten Element entspricht.

*stop\_time*

[in] Barzeit, die dem letzten Element entspricht.

*time\_array[]*

[out] Feld des Typs [datetime](#).

### Rückgabewert

Anzahl der kopierten Elemente des Feldes -1beim [Fehler](#).

### Bemerkung

Wenn das Interval der angeforderten Daten ausser den zugänglichen Daten auf dem Server ist, kehrt die Funktion -1 zurück. Falls die Daten ausserhalb [TERMINAL\\_MAXBARS](#) angefordert werden (maximale Anzahl der Bars auf dem Chart) gibt die Funktion auch -1 zurück.

Wenn die angeforderten Timeserien noch nicht gebildet sind oder müssen vom Server geladen werden, kehrt die Funktion sofort bei der Anforderung der Daten aus dem Indikator -1 zurück, aber dabei wird das Prozess der Ladung/Bildung initialisiert.

Bei der Anforderung der Daten aus dem Experten oder Script wird die [Ladung vom Server](#) initialisiert, wenn dieses Terminal keine lokale Daten hat oder Bildung der notwendigen Timeserie fängt an, wenn Daten aus der lokalen Geschichte gebildet werden kann, aber sie sind noch nicht fertig. Funktion kehrt diese Anzahl der Daten, die zum Augenblick des Timeoutsablaufs fertig sind, aber Ladung der Geschichte wird fortsetzen und bei der weiteren analogen Anforderung kehrt die Funktion schon mehr Daten zurück.

Bei der Anforderung der Daten nach Anfangsdatum und Anzahl der angeforderten Elemente kehren nur die Daten zurück, deren Datum kleiner (früher) oder gleich dem angegebenen Datum ist. Dabei

wird das Interval mit Genauigkeit bis zur Sekunde vorgegeben und berücksichtigt. D.h. die Zeit der Öffnung jeder Bar, für die der Wert zurückgegeben wird (Volumen, Spread, Wert im Indikatorpuffer, Preis Open, High, Low, Close oder Zeit der Öffnung Time), ist immer gleich oder kleiner als das angegebene Datum.

Bei der Anforderung der Daten im vorgegebenen Datenbereich kehren nur die Daten zurück, die in das angeforderte Interval kommen, dabei wird das Interval mit Genauigkeit bis zur Sekunde vorgegeben und berücksichtigt. D.h. die Zeit der Öffnung jeder Bar, für die der Wert zurückgegeben wird (Volumen, Spread, Wert im Indikatorpuffer, Preis Open, High, Low, Close oder Zeit der Öffnung Time) befindet sich immer im angeforderten Interval.

So wenn der laufende Wochentag Samstag ist, kehrt die Funktion beim Versuch, Daten in dem wöchentlichen Timeframe mit Andeutung *start\_time=Letzter\_Dienstag* und *stop\_time=Letzter\_Freitag* zu kopieren, 0 zurück, denn die Eröffnungszeit im wöchentlichen Timeframe ist immer der Sonntag, aber keine angegebene Bar gerät in den angegebenen Bereich.

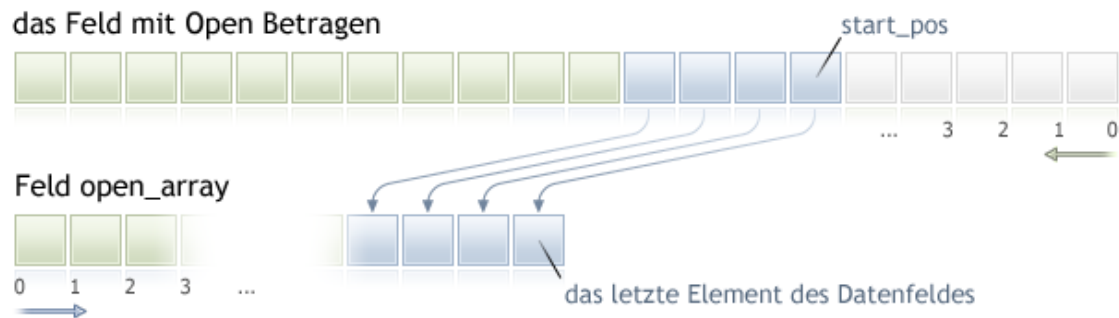
Wenn man muss, den ersten Wert zu erhalten, der der laufenden nicht beendeten Bar entspricht, kann man die erste Aufrufform mit der Andeutung *start\_pos=0* und *count=1* verwenden.

Sehen Sie das Beispiel von Anfrage der historischen Daten in [Methode der Objektbindung](#). Der Skript zeigt, wie Daten von Indikator [iFractals](#) an den letztem 1000 Balken zu erhalten und wie dann die letzten 10 Fraktale nach oben und 10 Fraktale nach unten auf dem Chart anzuzeigen. Sie können eine ähnliche Technik für alle Indikatoren, die fehlende Werte haben und typischerweise unter Verwendung der folgenden [Stil](#) hergestellt werden, verwenden:

- [DRAW\\_SECTION](#),
- [DRAW\\_ARROW](#),
- [DRAW\\_ZIGZAG](#),
- [DRAW\\_COLOR\\_SECTION](#),
- [DRAW\\_COLOR\\_ARROW](#),
- [DRAW\\_COLOR\\_ZIGZAG](#).

## CopyOpen

Funktion erhält im Feld `open_array` historische Daten der Eröffnungszeit der Bars für das angegebene Paar Symbol-Periode in der angegebenen Menge. Es muss bemerkt werden, dass Abzählen der Elemente von der Startposition von der Gegenwart zur Vergangenheit durchgeführt wird, d.h. die Anfangsposition, 0, bedeutet die laufende Bar.



Beim Kopieren der im voraus unbekannten Anzahl der Daten ist es empfehlenswert, als Feld-Rezipient ein [dynamisches Feld](#) zu verwenden, denn wenn die Anzahl der Daten weniger (oder mehr) ist, als ein Feld enthalten kann, versucht man das Feld so zu verteilen, dass die angeforderten Daten vollständig hineinpassen.

Wenn man die im voraus bekannte Anzahl der Daten kopieren muss, ist es besser dafür einen [statisch verteilten Puffer](#) zu verwenden, um unnötige Neuverteilung des Speichers zu vermeiden.

Es ist egal, welche Eigenschaft das Empfangsfeld hat - `as_series=true` oder `as_series=false`, Daten werden so kopiert, dass das älteste Element am Anfang des physischen Speichers sein wird, der für das Feld verteilt wurde. Es gibt 3 Varianten der Funktion.

### Aufruf nach Anfangsposition und Anzahl der angeforderten Elemente

```
int CopyOpen(
    string          symbol_name,      // Symbolname
    ENUM_TIMEFRAMES timeframe,        // Periode
    int             start_pos,         // Anfangsposition
    int             count,             // Anzahl für Kopieren
    double          open_array[]      // Feld für Kopieren der Eröffnungspreise
);
```

### Aufruf nach Anfangsdatum und Anzahl der angeforderten Elemente

```
int CopyOpen(
    string          symbol_name,      // Symbolname
    ENUM_TIMEFRAMES timeframe,        // Periode
    datetime        start_time,       // Anfangsdatum
    int             count,             // Anzahl für Kopieren
    double          open_array[]      // Feld für Kopieren der Eröffnungspreise
);
```

### Aufruf nach Anfangs- und Beendigungsdatum des angeforderten Zeitintervalls

```
int CopyOpen(  
    string          symbol_name,      // Symbolname  
    ENUM_TIMEFRAMES timeframe,       // Periode  
    datetime        start_time,      // Anfangsdatum  
    datetime        stop_time,       // Beendigungsdatum  
    double          open_array[]     // Feld für Kopieren der Eröffnungspreise  
);
```

### Parameter

*symbol\_name*

[in] Symbol.

*timeframe*

[in] Periode.

*start\_pos*

[in] Nummer des ersten kopierten Elementes.

*count*

[in] Anzahl der kopierten Elemente.

*start\_time*

[in] Zeit der Bar, die dem ersten Element entspricht.

*stop\_time*

[in] Barzeit, die dem letzten Element entspricht.

*open\_array[]*

[out] Feld des Typs [double](#).

### Rückgabewert

Anzahl der kopierten Elemente oder -1 beim [Fehler](#).

### Bemerkung

Wenn das Interval der angeforderten Daten ausser den zugänglichen Daten auf dem Server ist, kehrt die Funktion -1 zurück. Falls die Daten ausserhalb [TERMINAL\\_MAXBARS](#) angefordert werden (maximale Anzahl der Bars auf dem Chart) gibt die Funktion auch -1 zurück.

Wenn die angeforderten Timeserien noch nicht gebildet sind oder müssen vom Server geladen werden, kehrt die Funktion sofort bei der Anforderung der Daten aus dem Indikator -1 zurück, aber dabei wird das Prozess der Ladung/Bildung initialisiert.

Bei der Anforderung der Daten aus dem Experten oder Script wird die [Ladung vom Server](#) initialisiert, wenn dieses Terminal keine lokale Daten hat oder Bildung der notwendigen Timeserie fängt an, wenn Daten aus der lokalen Geschichte gebildet werden kann, aber sie sind noch nicht fertig. Funktion kehrt diese Anzahl der Daten, die zum Augenblick des Timeoutsablaufs fertig sind, aber Ladung der Geschichte wird fortsetzen und bei der weiteren analogen Anforderung kehrt die Funktion schon mehr Daten zurück.

Bei der Anforderung der Daten nach Anfangsdatum und Anzahl der angeforderten Elemente kehren nur die Daten zurück, deren Datum kleiner (früher) oder gleich dem angegebenen Datum ist. Dabei



wird das Interval mit Genauigkeit bis zur Sekunde vorgegeben und berücksichtigt. D.h. die Zeit der Öffnung jeder Bar, für die der Wert zurückgegeben wird (Volumen, Spread, Wert im Indikatorpuffer, Preis Open, High, Low, Close oder Zeit der Öffnung Time), ist immer gleich oder kleiner als das angegebene Datum.

Bei der Anforderung der Daten im vorgegebenen Datenbereich kehren nur die Daten zurück, die in das angeforderte Interval kommen, dabei wird das Interval mit Genauigkeit bis zur Sekunde vorgegeben und berücksichtigt. D.h. die Zeit der Öffnung jeder Bar, für die der Wert zurückgegeben wird (Volumen, Spread, Wert im Indikatorpuffer, Preis Open, High, Low, Close oder Zeit der Öffnung Time) befindet sich immer im angeforderten Interval.

So wenn der laufende Wochentag Samstag ist, kehrt die Funktion beim Versuch, Daten in dem wöchentlichen Timeframe mit Andeutung *start\_time=Letzter\_Dienstag* und *stop\_time=Letzter\_Freitag* zu kopieren, 0 zurück, denn die Eröffnungszeit im wöchentlichen Timeframe ist immer der Sonntag, aber keine angegebene Bar gerät in den angegebenen Bereich.

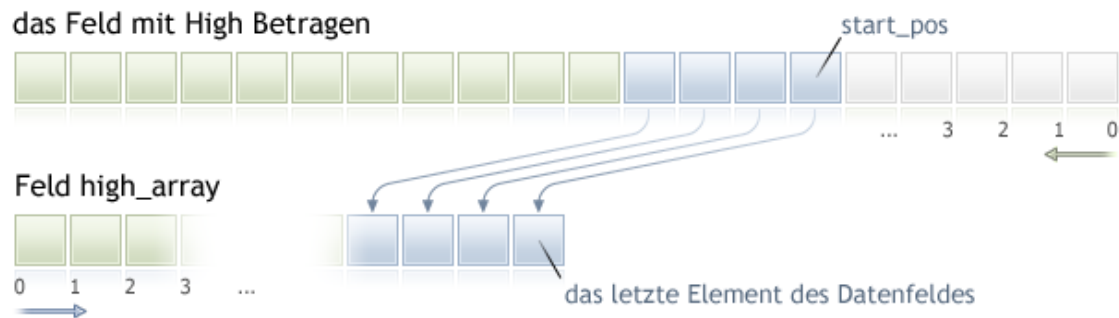
Wenn man muss, den ersten Wert zu erhalten, der der laufenden nicht beendeten Bar entspricht, kann man die erste Aufrufform mit der Andeutung *start\_pos=0* und *count=1* verwenden.

Sehen Sie das Beispiel von Anfrage der historischen Daten in [Methode der Objektbindung](#). Der Skript zeigt, wie Daten von Indikator [iFractals](#) an den letztem 1000 Balken zu erhalten und wie dann die letzten 10 Fraktale nach oben und 10 Fraktale nach unten auf dem Chart anzuzeigen. Sie können eine ähnliche Technik für alle Indikatoren, die fehlende Werte haben und typischerweise unter Verwendung der folgenden [Stil](#) hergestellt werden, verwenden:

- [DRAW\\_SECTION](#),
- [DRAW\\_ARROW](#),
- [DRAW\\_ZIGZAG](#),
- [DRAW\\_COLOR\\_SECTION](#),
- [DRAW\\_COLOR\\_ARROW](#),
- [DRAW\\_COLOR\\_ZIGZAG](#).

## CopyHigh

Funktion erhält im Feld `high_array` historische Daten der Eröffnungszeit der Bars für das angegebene Paar Symbol-Periode in der angegebenen Menge. Es muss bemerkt werden, dass Abzählen der Elemente von der Startposition von der Gegenwart zur Vergangenheit durchgeführt wird, d.h. die Anfangsposition, 0, bedeutet die laufende Bar.



Beim Kopieren der im voraus unbekannten Anzahl der Daten ist es empfehlenswert, als Feld-Rezipient ein [dynamisches Feld](#) zu verwenden, denn wenn die Anzahl der Daten weniger (oder mehr) ist, als ein Feld enthalten kann, versucht man das Feld so zu verteilen, dass die angeforderten Daten vollständig hineinpassen.

Wenn man die im voraus bekannte Anzahl der Daten kopieren muss, ist es besser dafür einen [statisch verteilten Puffer](#) zu verwenden, um unnötige Neuverteilung des Speichers zu vermeiden.

Es ist egal, welche Eigenschaft das Empfangsfeld hat - `as_series=true` oder `as_series=false`, Daten werden so kopiert, dass das älteste Element am Anfang des physischen Speichers sein wird, der für das Feld verteilt wurde. Es gibt 3 Varianten der Funktion.

### Aufruf nach Anfangsposition und Anzahl der angeforderten Elemente

```
int CopyHigh(
    string          symbol_name,      // Symbolname
    ENUM_TIMEFRAMES timeframe,        // Periode
    int             start_pos,         // Anfangsdatum
    int             count,             // Anzahl für Kopieren
    double          high_array[]       // Feld für Kopieren der maximalen Preise
);
```

### Aufruf nach Anfangsdatum und Anzahl der angeforderten Elemente

```
int CopyHigh(
    string          symbol_name,      // Symbolname
    ENUM_TIMEFRAMES timeframe,        // Periode
    datetime        start_time,       // Anfangsdatum
    int             count,             // Anzahl für Kopieren
    double          high_array[]       // Feld für Kopieren der maximalen Preise
);
```

### Aufruf nach Anfangs- und Beendigungsdatum des angeforderten Zeitintervalls

```

int CopyHigh(
    string          symbol_name,      // Symbolname
    ENUM_TIMEFRAMES timeframe,       // Periode
    datetime        start_time,      // Anfangsdatum
    datetime        stop_time,       // Beendigungsdatum
    double          high_array[]     // Feld für Kopieren der maximalen Preise
);

```

### Parameter

*symbol\_name*

[in] Symbol.

*timeframe*

[in] Periode.

*start\_pos*

[in] Nummer des ersten kopierten Elementes.

*count*

[in] Anzahl der kopierten Elemente.

*start\_time*

[in] Barzeit, die dem ersten Element entspricht.

*stop\_time*

[in] Barzeit, die dem letzten Element entspricht.

*high\_array[]*

[out] Feld des Typs [double](#).

### Rückgabewert

Anzahl der kopierten Elemente des Feldes oder -1 beim [Fehler](#).

### Bemerkung

Wenn das Interval der angeforderten Daten ausser den zugänglichen Daten auf dem Server ist, kehrt die Funktion -1 zurück. Falls die Daten ausserhalb [TERMINAL\\_MAXBARS](#) angefordert werden (maximale Anzahl der Bars auf dem Chart) gibt die Funktion auch -1 zurück.

Wenn die angeforderten Timeserien noch nicht gebildet sind oder müssen vom Server geladen werden, kehrt die Funktion sofort bei der Anforderung der Daten aus dem Indikator -1 zurück, aber dabei wird das Prozess der Ladung/Bildung initialisiert.

Bei der Anforderung der Daten aus dem Experten oder Script wird die [Ladung vom Server](#) initialisiert, wenn dieses Terminal keine lokale Daten hat oder Bildung der notwendigen Timeserie fängt an, wenn Daten aus der lokalen Geschichte gebildet werden kann, aber sie sind noch nicht fertig. Funktion kehrt diese Anzahl der Daten, die zum Augenblick des Timeoutsablaufs fertig sind, aber Ladung der Geschichte wird fortsetzen und bei der weiteren analogen Anforderung kehrt die Funktion schon mehr Daten zurück.

Bei der Anforderung der Daten nach Anfangsdatum und Anzahl der angeforderten Elemente kehren nur die Daten zurück, deren Datum kleiner (früher) oder gleich dem angegebenen Datum ist. Dabei

wird das Interval mit Genauigkeit bis zur Sekunde vorgegeben und berücksichtigt. D.h. die Zeit der Öffnung jeder Bar, für die der Wert zurückgegeben wird (Volumen, Spread, Wert im Indikatorpuffer, Preis Open, High, Low, Close oder Zeit der Öffnung Time), ist immer gleich oder kleiner als das angegebene Datum.

Bei der Anforderung der Daten im vorgegebenen Datenbereich kehren nur die Daten zurück, die in das angeforderte Interval kommen, dabei wird das Interval mit Genauigkeit bis zur Sekunde vorgegeben und berücksichtigt. D.h. die Zeit der Öffnung jeder Bar, für die der Wert zurückgegeben wird (Volumen, Spread, Wert im Indikatorpuffer, Preis Open, High, Low, Close oder Zeit der Öffnung Time) befindet sich immer im angeforderten Interval.

So wenn der laufende Wochentag Samstag ist, kehrt die Funktion beim Versuch, Daten in dem wöchentlichen Timeframe mit Andeutung *start\_time=Letzter\_Dienstag* und *stop\_time=Letzter\_Freitag* zu kopieren, 0 zurück, denn die Eröffnungszeit im wöchentlichen Timeframe ist immer der Sonntag, aber keine angegebene Bar gerät in den angegebenen Bereich.

Wenn man muss, den ersten Wert zu erhalten, der der laufenden nicht beendeten Bar entspricht, kann man die erste Aufrufform mit der Andeutung *start\_pos=0* und *count=1* verwenden.

#### Beispiel:

```
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Beispiel der Ausgabe der Werte High[i] und Low[i]"
#property description "für Bars, die zufaellig gewaehlt wurden"

double High[],Low[];

//+-----+
//| Bekommen wir Low für die angegebene Nummer der Bar |
//+-----+
double iLow(string symbol,ENUM_TIMEFRAMES timeframe,int index)
{
    double low=0;
    ArraySetAsSeries(Low,true);
    int copied=CopyLow(symbol,timeframe,0,Bars(symbol,timeframe),Low);
    if(copied>0 && index<copied) low=Low[index];
    return(low);
};

//+-----+
//| Bekommen wir High für die angegebene Nummer der Bar |
//+-----+
double iHigh(string symbol,ENUM_TIMEFRAMES timeframe,int index)
{
    double high=0;
    ArraySetAsSeries(High,true);
    int copied=CopyHigh(symbol,timeframe,0,Bars(symbol,timeframe),High);
    if(copied>0 && index<copied) high=High[index];
    return(high);
};
```

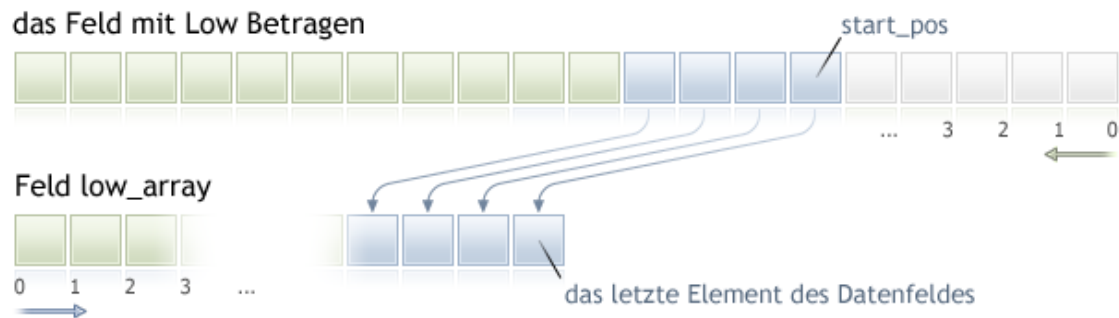
```
//+-----+
//| Expert tick function |
//+-----+
void OnTick()
{
//--- geben wir für jeden Tick die Werte High und Low für Bar mit Index aus,
//--- der einer Ticksekunde gleich ist
    datetime t=TimeCurrent();
    int sec=t%60;
    printf("High[%d] = %G Low[%d] = %G",
        sec,iHigh(Symbol(),0,sec),
        sec,iLow(Symbol(),0,sec));
};
```

Sehen Sie ein umfassenderes Beispiel von Anfrage der historischen Daten in [Methode der Objektbindung](#). Der Skript zeigt, wie Daten von Indikator [iFractals](#) an den letztem 1000 Balken zu erhalten und wie dann die letzten 10 Fraktale nach oben und 10 Fraktale nach unten auf dem Chart anzuzeigen. Sie können eine ähnliche Technik für alle Indikatoren, die fehlende Werte haben und typischerweise unter Verwendung der folgenden [Stil](#) hergestellt werden, verwenden:

- [DRAW\\_SECTION](#),
- [DRAW\\_ARROW](#),
- [DRAW\\_ZIGZAG](#),
- [DRAW\\_COLOR\\_SECTION](#),
- [DRAW\\_COLOR\\_ARROW](#),
- [DRAW\\_COLOR\\_ZIGZAG](#).

## CopyLow

Funktion erhält im Feld `low_array` historische Daten der minimalen Preise der Bars für das angegebene Paar Symbol-Periode in der angegebenen Menge. Es muss bemerkt werden, dass Abzählen der Elemente von der Startposition von der Gegenwart zur Vergangenheit durchgeführt wird, d.h. die Anfangsposition, 0, bedeutet die laufende Bar. .



Beim Kopieren der im voraus unbekannten Anzahl der Daten ist es empfehlenswert, als Feld-Rezipient ein [dynamisches Feld](#) zu verwenden, denn wenn die Anzahl der Daten weniger (oder mehr) ist, als ein Feld enthalten kann, versucht man das Feld so zu verteilen, dass die angeforderten Daten vollständig hineinpassen.

Wenn man die im voraus bekannte Anzahl der Daten kopieren muss, ist es besser dafür einen [statisch verteilten Puffer](#) zu verwenden, um unnötige Neuverteilung des Speichers zu vermeiden.

Es ist egal, welche Eigenschaft das Feld-Rezipient hat - `as_series=true` oder `as_series=false`, Daten werden so kopiert, dass das älteste Element am Anfang des physischen Speichers sein wird, der für das Feld verteilt wurde. Es gibt 3 Varianten der Funktion.

### Aufruf nach Anfangsposition und Anzahl der angeforderten Elemente

```
int CopyLow(
    string          symbol_name,      // Symbolname
    ENUM_TIMEFRAMES timeframe,        // Periode
    int             start_pos,        // Anfangsposition
    int             count,            // Anzahl für Kopieren
    double          low_array[]       // Feld für Kopieren der minimalen Preise
);
```

### Aufruf nach Anfangsdatum und Anzahl der angeforderten Elemente

```
int CopyLow(
    string          symbol_name,      // Symbolname
    ENUM_TIMEFRAMES timeframe,        // Periode
    datetime        start_time,       // Anfangsdatum
    int             count,            // Anzahl für Kopieren
    double          low_array[]       // Feld für Kopieren der minimalen Preise
);
```

### Aufruf nach Anfangs- und Beendigungsdatum des angeforderten Zeitintervalls

```

int CopyLow(
    string          symbol_name,      // Symbolname
    ENUM_TIMEFRAMES timeframe,       // Periode
    datetime        start_time,      // Anfangsdatum
    datetime        stop_time,       // Beendigungsdatum
    double          low_array[]      // Feld für Kopieren der minimalen Preise
);

```

### Parameter

*symbol\_name*

[in] Symbol.

*timeframe*

[in] Periode.

*start\_pos*

[in] Nummer des ersten kopierten Elementes.

*count*

[in] Anzahl der kopierten Elemente.

*start\_time*

[in] Barzeit, die dem ersten Element entspricht.

*stop\_time*

[in] Barzeit, die dem letzten Element entspricht.

*low\_array[]*

[out] Feld des Typs [double](#).

### Rückgabewert

Anzahl der kopierten Elemente des Feldes oder -1 beim [Fehler](#).

### Bemerkung

Wenn das Interval der angeforderten Daten ausser den zugänglichen Daten auf dem Server ist, kehrt die Funktion -1 zurück. Falls die Daten ausserhalb [TERMINAL\\_MAXBARS](#) angefordert werden (maximale Anzahl der Bars auf dem Chart) gibt die Funktion auch -1 zurück.

Wenn die angeforderten Timeserien noch nicht gebildet sind oder müssen vom Server geladen werden, kehrt die Funktion sofort bei der Anforderung der Daten aus dem Indikator -1 zurück, aber dabei wird das Prozess der Ladung/Bildung initialisiert.

Bei der Anforderung der Daten aus dem Experten oder Script wird die [Ladung vom Server](#) initialisiert, wenn dieses Terminal keine lokale Daten hat oder Bildung der notwendigen Timeserie fängt an, wenn Daten aus der lokalen Geschichte gebildet werden kann, aber sie sind noch nicht fertig. Funktion kehrt diese Anzahl der Daten, die zum Augenblick des Timeoutsablaufs fertig sind, aber Ladung der Geschichte wird fortsetzen und bei der weiteren analogen Anforderung kehrt die Funktion schon mehr Daten zurück.

Bei der Anforderung der Daten nach Anfangsdatum und Anzahl der angeforderten Elemente kehren nur die Daten zurück, deren Datum kleiner (früher) oder gleich dem angegebenen Datum ist. Dabei

wird das Interval mit Genauigkeit bis zur Sekunde vorgegeben und berücksichtigt. D.h. die Zeit der Öffnung jeder Bar, für die der Wert zurückgegeben wird (Volumen, Spread, Wert im Indikatorpuffer, Preis Open, High, Low, Close oder Zeit der Öffnung Time), ist immer gleich oder kleiner als das angegebene Datum.

Bei der Anforderung der Daten im vorgegebenen Datenbereich kehren nur die Daten zurück, die in das angeforderte Interval kommen, dabei wird das Interval mit Genauigkeit bis zur Sekunde vorgegeben und berücksichtigt. D.h. die Zeit der Öffnung jeder Bar, für die der Wert zurückgegeben wird (Volumen, Spread, Wert im Indikatorpuffer, Preis Open, High, Low, Close oder Zeit der Öffnung Time) befindet sich immer im angeforderten Interval.

So wenn der laufende Wochentag Samstag ist, kehrt die Funktion beim Versuch, Daten in dem wöchentlichen Timeframe mit Andeutung *start\_time=Letzter\_Dienstag* und *stop\_time=Letzter\_Freitag* zu kopieren, 0 zurück, denn die Eröffnungszeit im wöchentlichen Timeframe ist immer der Sonntag, aber keine angegebene Bar gerät in den angegebenen Bereich.

Wenn man muss, den ersten Wert zu erhalten, der der laufenden nicht beendeten Bar entspricht, kann man die erste Aufrufform mit der Andeutung *start\_pos=0* und *count=1* verwenden.

Sehen Sie das Beispiel von Anfrage der historischen Daten in [Methode der Objektbindung](#). Der Skript zeigt, wie Daten von Indikator [iFractals](#) an den letztem 1000 Balken zu erhalten und wie dann die letzten 10 Fraktale nach oben und 10 Fraktale nach unten auf dem Chart anzuzeigen. Sie können eine ähnliche Technik für alle Indikatoren, die fehlende Werte haben und typischerweise unter Verwendung der folgenden [Stil](#) hergestellt werden, verwenden:

- [DRAW\\_SECTION](#),
- [DRAW\\_ARROW](#),
- [DRAW\\_ZIGZAG](#),
- [DRAW\\_COLOR\\_SECTION](#),
- [DRAW\\_COLOR\\_ARROW](#),
- [DRAW\\_COLOR\\_ZIGZAG](#).

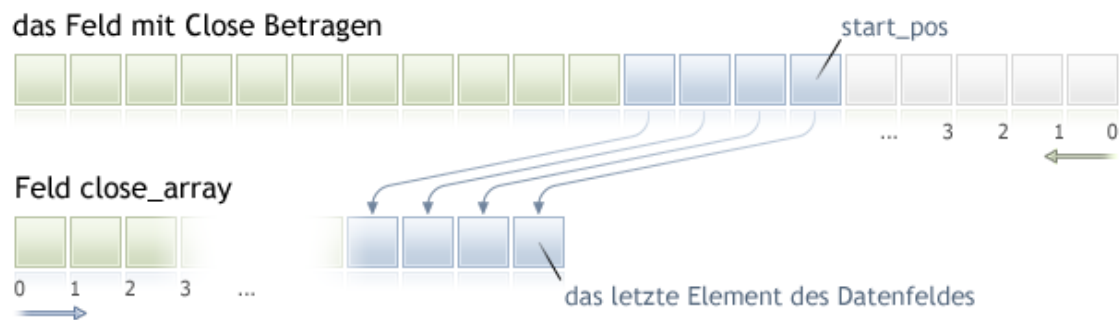
Sehen Sie auch

[CopyHigh](#)



## CopyClose

Funktion erhält im Feld `close_array` historische Daten der minimalen Preisen der Bars für das angegebene Paar Symbol-Periode in der angegebenen Menge. Es muss bemerkt werden, dass Abzählen der Elemente von der Startposition von der Gegenwart zur Vergangenheit durchgeführt wird, d.h. die Anfangsposition, 0, bedeutet die laufende Bar.



Beim Kopieren der im voraus unbekannten Anzahl der Daten ist es empfehlenswert, als Feld-Rezipient ein [dynamisches Feld](#), zu verwenden, denn wenn die Anzahl der Daten weniger (oder mehr) ist, als ein Feld enthalten kann, versucht man das Feld so zu verteilen, dass die angeforderten Daten vollständig hineinpassen.

Wenn man die im voraus bekannte Anzahl der Daten kopieren muss, ist es besser dafür einen [statisch verteilten Puffer](#) zu verwenden, um unnötige Neuverteilung des Speichers zu vermeiden.

Es ist egal, welche Eigenschaft das Feld-Rezipient hat - `as_series=true` oder `as_series=false`, Daten werden so kopiert, dass das älteste Element am Anfang des physischen Speichers sein wird, der für das Feld verteilt wurde. Es gibt 3 Varianten der Funktion..

### Aufruf nach Anfangsposition und Anzahl der angeforderten Elemente

```
int CopyClose(
    string          symbol_name,      // Symbolname
    ENUM_TIMEFRAMES timeframe,        // Periode
    int             start_pos,         // Startposition
    int             count,             // Anzahl für Kopieren
    double          close_array[]     // Feld für Kopieren der Schlusspreise
);
```

### Aufruf nach Anfangsdatum und Anzahl der angeforderten Elemente

```
int CopyClose(
    string          symbol_name,      // Symbolname
    ENUM_TIMEFRAMES timeframe,        // Periode
    datetime        start_time,       // Anfangsdatum
    int             count,             // Anzahl für Kopieren
    double          close_array[]     // Feld für Kopieren der Schlusspreise
);
```

### Aufruf nach Anfangs- und Beendigungsdatum des angeforderten Zeitintervalls

```

int CopyClose (
    string          symbol_name,      // Symbolname
    ENUM_TIMEFRAMES timeframe,       // Periode
    datetime        start_time,      // Anfangsdatum
    datetime        stop_time,       // Beendigungsdatum
    double          close_array[]    // Feld für Kopieren der Schlusspreise
);

```

### Parameter

*symbol\_name*

[in] Symbol.

*timeframe*

[in] Periode.

*start\_pos*

[in] Nummer des ersten kopierten Elementes.

*count*

[in] Anzahl der kopierten Elemente.

*start\_time*

[in] Barzeit, die dem ersten Element entspricht.

*stop\_time*

[in] Barzeit, die dem letzten Element entspricht.

*close\_array[]*

[out] Feld des Typs [double](#).

### Rückgabewert

Anzahl der kopierten Elemente des Feldes oder -1 beim [Fehler](#).

### Bemerkung

Wenn das Interval der angeforderten Daten ausser den zugänglichen Daten auf dem Server ist, kehrt die Funktion -1 zurück. Falls die Daten ausserhalb [TERMINAL\\_MAXBARS](#) angefordert werden (maximale Anzahl der Bars auf dem Chart) gibt die Funktion auch -1 zurück.

Wenn die angeforderten Timeserien noch nicht gebildet sind oder müssen vom Server geladen werden, kehrt die Funktion sofort bei der Anforderung der Daten aus dem Indikator -1 zurück, aber dabei wird das Prozess der Ladung/Bildung initialisiert.

Bei der Anforderung der Daten aus dem Experten oder Script wird die [Ladung vom Server](#) initialisiert, wenn dieses Terminal keine lokale Daten hat oder Bildung der notwendigen Timeserie fängt an, wenn Daten aus der lokalen Geschichte gebildet werden kann, aber sie sind noch nicht fertig. Funktion kehrt diese Anzahl der Daten, die zum Augenblick des Timeoutsablaufs fertig sind, aber Ladung der Geschichte wird fortsetzen und bei der weiteren analogen Anforderung kehrt die Funktion schon mehr Daten zurück.

Bei der Anforderung der Daten nach Anfangsdatum und Anzahl der angeforderten Elemente kehren nur die Daten zurück, deren Datum kleiner (früher) oder gleich dem angegebenen Datum ist. Dabei

wird das Interval mit Genauigkeit bis zur Sekunde vorgegeben und berücksichtigt. D.h. die Zeit der Öffnung jeder Bar, für die der Wert zurückgegeben wird (Volumen, Spread, Wert im Indikatorpuffer, Preis Open, High, Low, Close oder Zeit der Öffnung Time), ist immer gleich oder kleiner als das angegebene Datum.

Bei der Anforderung der Daten im vorgegebenen Datenbereich kehren nur die Daten zurück, die in das angeforderte Interval kommen, dabei wird das Interval mit Genauigkeit bis zur Sekunde vorgegeben und berücksichtigt. D.h. die Zeit der Öffnung jeder Bar, für die der Wert zurückgegeben wird (Volumen, Spread, Wert im Indikatorpuffer, Preis Open, High, Low, Close oder Zeit der Öffnung Time) befindet sich immer im angeforderten Interval.

So wenn der laufende Wochentag Samstag ist, kehrt die Funktion beim Versuch, Daten in dem wöchentlichen Timeframe mit Andeutung *start\_time=Letzter\_Dienstag* und *stop\_time=Letzter\_Freitag* zu kopieren, 0 zurück, denn die Eröffnungszeit im wöchentlichen Timeframe ist immer der Sonntag, aber keine angegebene Bar gerät in den angegebenen Bereich.

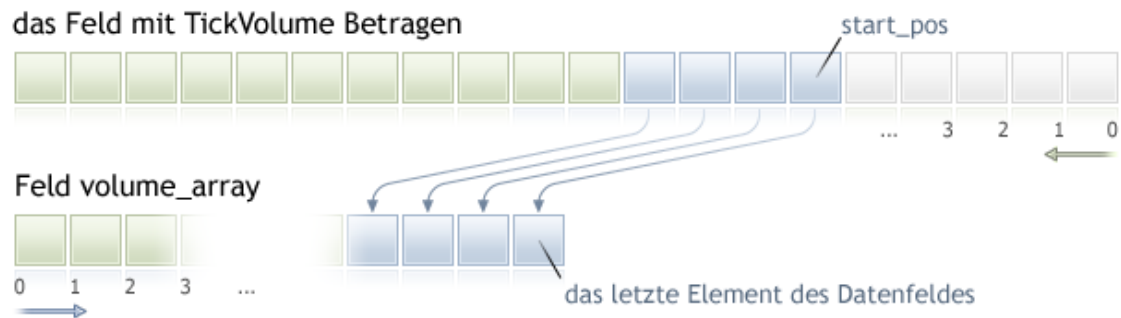
Wenn man muss, den ersten Wert zu erhalten, der der laufenden nicht beendeten Bar entspricht, kann man die erste Aufrufform mit der Andeutung *start\_pos=0* und *count=1* verwenden.

Sehen Sie das Beispiel von Anfrage der historischen Daten in [Methode der Objektbindung](#). Der Skript zeigt, wie Daten von Indikator [iFractals](#) an den letztem 1000 Balken zu erhalten und wie dann die letzten 10 Fraktale nach oben und 10 Fraktale nach unten auf dem Chart anzuzeigen. Sie können eine ähnliche Technik für alle Indikatoren, die fehlende Werte haben und typischerweise unter Verwendung der folgenden [Stil](#) hergestellt werden, verwenden:

- [DRAW\\_SECTION](#),
- [DRAW\\_ARROW](#),
- [DRAW\\_ZIGZAG](#),
- [DRAW\\_COLOR\\_SECTION](#),
- [DRAW\\_COLOR\\_ARROW](#),
- [DRAW\\_COLOR\\_ZIGZAG](#).

## CopyTickVolume

Funktion erhält im Feld `volume_array` historische Daten der minimalen Preisen der Bars für das angegebene Paar Symbol-Periode in der angegebenen Menge. Es muss bemerkt werden, dass Abzählen der Elemente von der Startposition von der Gegenwart zur Vergangenheit durchgeführt wird, d.h. die Anfangsposition, 0, bedeutet die laufende Bar.



Beim Kopieren der im voraus unbekannten Anzahl der Daten ist es empfehlenswert, als Feld-Rezipient ein [dynamisches Feld](#) zu verwenden, denn wenn die Anzahl der Daten weniger (oder mehr) ist, als ein Feld enthalten kann, versucht man das Feld so zu verteilen, dass die angeforderten Daten vollstaendig hineinpassen.

Wenn man die im voraus bekannte Anzahl der Daten kopieren muss, ist es besser dafür einen [statisch verteilten Puffer](#), um unnoetige Neuverteilung des Speichers zu vermeiden.

Es ist egal, welche Eigenschaft das Feld-Rezipient hat - `as_series=true` oder `as_series=false`, Daten werden so kopiert, dass das aelteste Element am Anfang des physischen Speichers sein wird, der für das Feld verteilt wurde. Es gibt 3 Varianten der Funktion.

### Aufruf nach Anfangsdatum und Anzahl der angeforderten Elemente

```
int CopyTickVolume(
    string          symbol_name,      // Symbolname
    ENUM_TIMEFRAMES timeframe,        // Periode
    int             start_pos,        // Anfangsposition
    int             count,            // Anzahl für Kopieren
    long            volume_array[]    // Feld für Kopieren der Tickvolumen
);
```

### Aufruf nach Anfangsdatum und Anzahl der angeforderten Elemente

```
int CopyTickVolume(
    string          symbol_name,      // Symbolname
    ENUM_TIMEFRAMES timeframe,        // Periode
    datetime        start_time,       // Anfangsdatum
    int             count,            // Anzahl für Kopieren
    long            volume_array[]    // Feld für Kopieren der Tickvolumen
);
```

### Aufruf nach Anfangs- und Beendigungsdatum des angeforderten Zeitintervalls

```
int CopyTickVolume(
```

```

string      symbol_name,      // Symbolname
ENUM_TIMEFRAMES timeframe,    // Periode
datetime    start_time,       // Anfangsdatum
datetime    stop_time,        // Beendigungsdatum
long        volume_array[]    // Feld für Kopieren der Tickvolumen
);

```

### Parameter

*symbol\_name*

[in] Symbol.

*timeframe*

[in] Periode.

*start\_pos*

[in] Nummer des ersten kopierten Elementes.

*count*

[in] Anzahl der kopierten Elemente.

*start\_time*

[in] Barzeit, die dem ersten Element entspricht.

*stop\_time*

[in] Barzeit, die dem letzten Element entspricht.

*volume\_array[]*

[out] Feld des Typs [long](#).

### Rückgabewert

Anzahl der kopierten Elemente des Feldes oder -1 beim [Fehler](#).

### Bemerkung

Wenn das Interval der angeforderten Daten ausser den zugänglichen Daten auf dem Server ist, kehrt die Funktion -1 zurück. Falls die Daten ausserhalb [TERMINAL\\_MAXBARS](#) angefordert werden (maximale Anzahl der Bars auf dem Chart) gibt die Funktion auch -1 zurück.

Wenn die angeforderten Timeserien noch nicht gebildet sind oder müssen vom Server geladen werden, kehrt die Funktion sofort bei der Anforderung der Daten aus dem Indikator -1 zurück, aber dabei wird das Prozess der Ladung/Bildung initialisiert.

Bei der Anforderung der Daten aus dem Experten oder Script wird die [Ladung vom Server](#) initialisiert, wenn dieses Terminal keine lokale Daten hat oder Bildung der notwendigen Timeserie fängt an, wenn Daten aus der lokalen Geschichte gebildet werden kann, aber sie sind noch nicht fertig. Funktion kehrt diese Anzahl der Daten, die zum Augenblick des Timeoutsablaufs fertig sind, aber Ladung der Geschichte wird fortsetzen und bei der weiteren analogen Anforderung kehrt die Funktion schon mehr Daten zurück.

Bei der Anforderung der Daten nach Anfangsdatum und Anzahl der angeforderten Elemente kehren nur die Daten zurück, deren Datum kleiner (früher) oder gleich dem angegebenen Datum ist. Dabei wird das Interval mit Genauigkeit bis zur Sekunde vorgegeben und berücksichtigt. D.h. die Zeit der

Öffnung jeder Bar, für die der Wert zurückgegeben wird (Volumen, Spread, Wert im Indikatorpuffer, Preis Open, High, Low, Close oder Zeit der Öffnung Time), ist immer gleich oder kleiner als das angegebene Datum.

Bei der Anforderung der Daten im vorgegebenen Datenbereich kehren nur die Daten zurück, die in das angeforderte Interval kommen, dabei wird das Interval mit Genauigkeit bis zur Sekunde vorgegeben und berücksichtigt. D.h. die Zeit der Öffnung jeder Bar, für die der Wert zurückgegeben wird (Volumen, Spread, Wert im Indikatorpuffer, Preis Open, High, Low, Close oder Zeit der Öffnung Time) befindet sich immer im angeforderten Interval.

So wenn der laufende Wochentag Samstag ist, kehrt die Funktion beim Versuch, Daten in dem wöchentlichen Timeframe mit Andeutung *start\_time=Letzter\_Dienstag* und *stop\_time=Letzter\_Freitag* zu kopieren, 0 zurück, denn die Eröffnungszeit im wöchentlichen Timeframe ist immer der Sonntag, aber keine angegebene Bar gerät in den angegebenen Bereich.

Wenn man muss, den ersten Wert zu erhalten, der der laufenden nicht beendeten Bar entspricht, kann man die erste Aufrufform mit der Andeutung *start\_pos=0* und *count=1* verwenden.

#### Beispiel:

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots 1
//---- plot TickVolume
#property indicator_label1 "TickVolume"
#property indicator_type1 DRAW_HISTOGRAM
#property indicator_color1 C'143,188,139'
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//--- input parameters
input int bars=3000;
//--- indicator buffers
double TickVolumeBuffer[];
//+-----+
//| Custom indicator initialization function |
//+-----+
void OnInit()
{
//--- indicator buffers mapping
SetIndexBuffer(0,TickVolumeBuffer,INDICATOR_DATA);
IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
; }
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
```

```

        const double &low[],
        const double &close[],
        const long &tick_volume[],
        const long &volume[],
        const int &spread[])
    {
//---
        if(prev_calculated==0)
        {
            long timeseries[];
            ArraySetAsSeries(timeseries,true);
            int prices=CopyTickVolume(Symbol(),0,0,bars,timeseries);
            for(int i=0;i<rates_total-prices;i++) TickVolumeBuffer[i]=0.0;
            for(int i=0;i<prices;i++) TickVolumeBuffer[rates_total-1-i]=timeseries[prices-1-i];
            Print("Wir haben erhalten die folgende Anzahl der historischen Werte TickVolume: ",prices);
        }
        else
        {
            long timeseries[];
            int prices=CopyTickVolume(Symbol(),0,0,1,timeseries);
            TickVolumeBuffer[rates_total-1]=timeseries[0];
        }
//--- return value of prev_calculated for next call
        return(rates_total);
    }

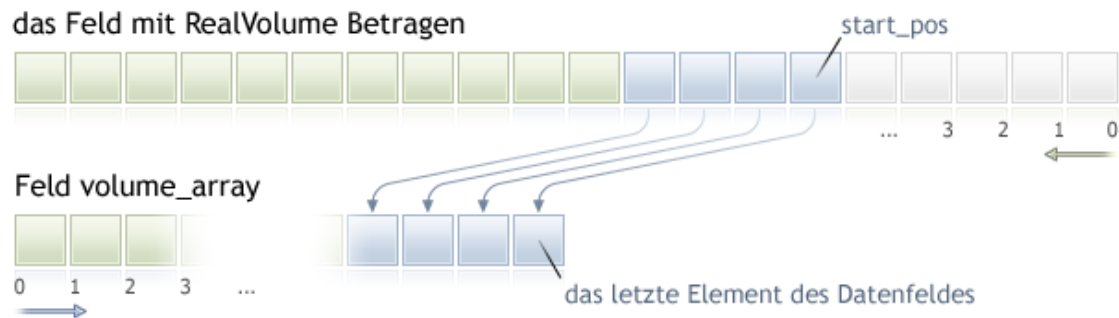
```

Sehen Sie ein umfassenderes Beispiel von Anfrage der historischen Daten in [Methode der Objektbindung](#). Der Skript zeigt, wie Daten von Indikator [iFractals](#) an den letztem 1000 Balken zu erhalten und wie dann die letzten 10 Fraktale nach oben und 10 Fraktale nach unten auf dem Chart anzuzeigen. Sie können eine ähnliche Technik für alle Indikatoren, die fehlende Werte haben und typischerweise unter Verwendung der folgenden [Stil](#) hergestellt werden, verwenden:

- [DRAW\\_SECTION](#),
- [DRAW\\_ARROW](#),
- [DRAW\\_ZIGZAG](#),
- [DRAW\\_COLOR\\_SECTION](#),
- [DRAW\\_COLOR\\_ARROW](#),
- [DRAW\\_COLOR\\_ZIGZAG](#).

## CopyRealVolume

Funktion erhält im Feld `volume_array` historische Daten der minimalen Preisen der Bars für das angegebene Paar Symbol-Periode in der angegebenen Menge. Es muss bemerkt werden, dass Abzählen der Elemente von der Startposition von der Gegenwart zur Vergangenheit durchgeführt wird, d.h. die Anfangsposition, 0, bedeutet die laufende Bar.



Beim Kopieren der im voraus unbekannten Anzahl der Daten ist es empfehlenswert, als Feld-Rezipient ein [dynamisches Feld](#) zu verwenden, denn wenn die Anzahl der Daten weniger (oder mehr) ist, als ein Feld enthalten kann, versucht man das Feld so zu verteilen, dass die angeforderten Daten vollständig hineinpassen.

Wenn man die im voraus bekannte Anzahl der Daten kopieren muss, ist es besser dafür einen [statisch verteilten Puffer](#) zu verwenden, um unnötige Umverteilung des Speichers zu vermeiden.

Es ist egal, welche Eigenschaft das Empfangsfeld hat - `as_series=true` oder `as_series=false`, Daten werden so kopiert, dass das älteste Element am Anfang des physischen Speichers sein wird, der für das Feld verteilt wurde. Es gibt 3 Varianten der Funktion.

### Aufruf nach Anfangsposition und Anzahl der angeforderten Elemente

```
int CopyRealVolume(
    string          symbol_name,      // Symbolname
    ENUM_TIMEFRAMES timeframe,        // Periode
    int             start_pos,         // Anfangsposition
    int             count,             // Anzahl für Kopieren
    long            volume_array[]     // Feld für Kopieren der Volumen
);
```

### Aufruf nach Anfangsdatum und Anzahl der angeforderten Elemente

```
int CopyRealVolume(
    string          symbol_name,      // Symbolname
    ENUM_TIMEFRAMES timeframe,        // Periode
    datetime        start_time,       // Anfangsdatum
    int             count,             // Anzahl für Kopieren
    long            volume_array[]     // Feld für Kopieren der Volumen
);
```

### Aufruf nach Anfangs- und Beendigungsdatum des angeforderten Zeitintervalls



```

int CopyRealVolume (
    string          symbol_name,      // Symbolname
    ENUM_TIMEFRAMES timeframe,       // Periode
    datetime        start_time,      // Anfangsdatum
    datetime        stop_time,       // Beendigungsdatum
    long            volume_array[]    // Feld für Kopieren der Volumen
);

```

### Parameter

*symbol\_name*

[in] Symbol.

*timeframe*

[in] Periode.

*start\_pos*

[in] Nummer des ersten kopierten Elementes.

*count*

[in] Anzahl der kopierten Elemente.

*start\_time*

[in] Barzeit, die dem ersten Element entspricht.

*stop\_time*

[in] Barzeit, die dem letzten Element entspricht.

*volume\_array[]*

[out] Feld des Typs [long](#).

### Rückgabewert

Anzahl der kopierten Elemente des Feldes oder -1 beim [Fehler](#).

### Bemerkung

Wenn das Interval der angeforderten Daten ausser den zugänglichen Daten auf dem Server ist, kehrt die Funktion -1 zurück. Falls die Daten ausserhalb [TERMINAL\\_MAXBARS](#) angefordert werden (maximale Anzahl der Bars auf dem Chart) gibt die Funktion auch -1 zurück.

Wenn die angeforderten Timeserien noch nicht gebildet sind oder müssen vom Server geladen werden, kehrt die Funktion sofort bei der Anforderung der Daten aus dem Indikator -1 zurück, aber dabei wird das Prozess der Ladung/Bildung initialisiert.

Bei der Anforderung der Daten aus dem Experten oder Script wird die [Ladung vom Server](#) initialisiert, wenn dieses Terminal keine lokale Daten hat oder Bildung der notwendigen Timeserie fängt an, wenn Daten aus der lokalen Geschichte gebildet werden kann, aber sie sind noch nicht fertig. Funktion kehrt diese Anzahl der Daten, die zum Augenblick des Timeoutsablaufs fertig sind, aber Ladung der Geschichte wird fortsetzen und bei der weiteren analogen Anforderung kehrt die Funktion schon mehr Daten zurück.

Bei der Anforderung der Daten nach Anfangsdatum und Anzahl der angeforderten Elemente kehren nur die Daten zurück, deren Datum kleiner (früher) oder gleich dem angegebenen Datum ist. Dabei

wird das Interval mit Genauigkeit bis zur Sekunde vorgegeben und berücksichtigt. D.h. die Zeit der Öffnung jeder Bar, für die der Wert zurückgegeben wird (Volumen, Spread, Wert im Indikatorpuffer, Preis Open, High, Low, Close oder Zeit der Öffnung Time), ist immer gleich oder kleiner als das angegebene Datum.

Bei der Anforderung der Daten im vorgegebenen Datenbereich kehren nur die Daten zurück, die in das angeforderte Interval kommen, dabei wird das Interval mit Genauigkeit bis zur Sekunde vorgegeben und berücksichtigt. D.h. die Zeit der Öffnung jeder Bar, für die der Wert zurückgegeben wird (Volumen, Spread, Wert im Indikatorpuffer, Preis Open, High, Low, Close oder Zeit der Öffnung Time) befindet sich immer im angeforderten Interval.

So wenn der laufende Wochentag Samstag ist, kehrt die Funktion beim Versuch, Daten in dem wöchentlichen Timeframe mit Andeutung *start\_time=Letzter\_Dienstag* und *stop\_time=Letzter\_Freitag* zu kopieren, 0 zurück, denn die Eröffnungszeit im wöchentlichen Timeframe ist immer der Sonntag, aber keine angegebene Bar gerät in den angegebenen Bereich.

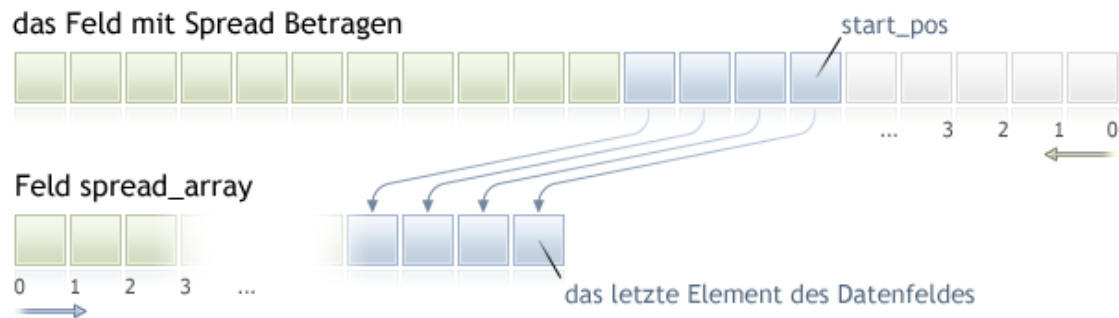
Wenn man muss, den ersten Wert zu erhalten, der der laufenden nicht beendeten Bar entspricht, kann man die erste Aufrufform mit der Andeutung *start\_pos=0* und *count=1* verwenden.

Sehen Sie das Beispiel von Anfrage der historischen Daten in [Methode der Objektbindung](#). Der Skript zeigt, wie Daten von Indikator [iFractals](#) an den letztem 1000 Balken zu erhalten und wie dann die letzten 10 Fraktale nach oben und 10 Fraktale nach unten auf dem Chart anzuzeigen. Sie können eine ähnliche Technik für alle Indikatoren, die fehlende Werte haben und typischerweise unter Verwendung der folgenden [Stil](#) hergestellt werden, verwenden:

- [DRAW\\_SECTION](#),
- [DRAW\\_ARROW](#),
- [DRAW\\_ZIGZAG](#),
- [DRAW\\_COLOR\\_SECTION](#),
- [DRAW\\_COLOR\\_ARROW](#),
- [DRAW\\_COLOR\\_ZIGZAG](#).

## CopySpread

Funktion bekommt im Feld `spread_array` historische Daten der Spreads für das angegebene Paar Symbol-Periode in der angegebenen Menge. Es muss bemerkt werden, dass Abzählen der Elemente von der Startposition von der Gegenwart zur Vergangenheit durchgeführt wird, d.h. die Anfangsposition, 0, bedeutet die laufende Bar.



Bei Kopieren der im voraus unbekannten Anzahl von Daten ist es empfehlenswert, als Feld-Rezipient ein [dynamisches Feld](#) zu verwenden, denn wenn die Anzahl der Daten weniger (oder mehr) ist, als ein Feld enthalten kann, versucht man das Feld so zu verteilen, dass die angeforderten Daten vollstaendig hineinpassen.

Wenn man die im voraus bekannte Anzahl der Daten kopieren muss, ist es besser dafür einen [statisch verteilten Puffer](#) zu verwenden, um unnoetige Neuverteilung des Speichers zu vermeiden.

Es ist egal, welche Eigenschaft das Empfangsfeld hat - `as_series=true` oder `as_series=false`, Daten werden so kopiert, dass das aelteste Element am Anfang des physischen Speichers sein wird, der für das Feld verteilt wurde. Es gibt 3 Varianten der Funktion.

### Aufruf nach Anfangsposition und Anzahl der angeforderten Elemente

```
int CopySpread(
    string          symbol_name,      // Symbolname
    ENUM_TIMEFRAMES timeframe,        // Periode
    int             start_pos,         // Anfangsposition
    int             count,             // Zahl für Kopieren
    int             spread_array[]     // Feld für Kopieren der Spreads
);
```

### Aufruf nach Anfangsdatum und Anzahl der angeforderten Elemente

```
int CopySpread(
    string          symbol_name,      // Symbolname
    ENUM_TIMEFRAMES timeframe,        // Periode
    datetime        start_time,       // Anfangsdatum
    int             count,             // Anzahl für Kopieren
    int             spread_array[]     // Feld für Kopieren der Spreads
);
```

### Aufruf nach Anfangs- und Beendigungsdatum des angeforderten Zeitintervalls

```

int CopySpread(
    string      symbol_name,      // Symbolname
    ENUM_TIMEFRAMES timeframe,    // Periode
    datetime    start_time,       // Anfangsdatum
    datetime    stop_time,        // Beendigungsdatum
    int         spread_array[]    // Feld für Kopieren der Spreads
);

```

### Parameter

*symbol\_name*

[in] Symbol.

*timeframe*

[in] Periode

*start\_pos*

[in] Nummer des ersten kopierten Elementes.

*count*

[in] Anzahl der kopierten Elemente.

*start\_time*

[in] Barzeit, die dem ersten Element entspricht.

*stop\_time*

[in] Barzeit, die dem letzten Element entspricht.

*spread\_array[]*

[out] Feld des Typs [int](#).

### Rückgabewert

Anzahl der kopierten Elemente des Feldes oder -1 beim [Fehler](#).

### Bemerkung

Wenn das Interval der angeforderten Daten ausser den zugänglichen Daten auf dem Server ist, kehrt die Funktion -1 zurück. Falls die Daten ausserhalb [TERMINAL\\_MAXBARS](#) angefordert werden (maximale Anzahl der Bars auf dem Chart) gibt die Funktion auch -1 zurück.

Wenn die angeforderten Timeserien noch nicht gebildet sind oder müssen vom Server geladen werden, kehrt die Funktion sofort bei der Anforderung der Daten aus dem Indikator -1 zurück, aber dabei wird das Prozess der Ladung/Bildung initialisiert.

Bei der Anforderung der Daten aus dem Experten oder Script wird die [Ladung vom Server](#) initialisiert, wenn dieses Terminal keine lokale Daten hat oder Bildung der notwendigen Timeserie fängt an, wenn Daten aus der lokalen Geschichte gebildet werden kann, aber sie sind noch nicht fertig. Funktion kehrt diese Anzahl der Daten, die zum Augenblick des Timeoutsablaufs fertig sind, aber Ladung der Geschichte wird fortsetzen und bei der weiteren analogen Anforderung kehrt die Funktion schon mehr Daten zurück.

Bei der Anforderung der Daten nach Anfangsdatum und Anzahl der angeforderten Elemente kehren nur die Daten zurück, deren Datum kleiner (früher) oder gleich dem angegebenen Datum ist. Dabei

wird das Intervall mit Genauigkeit bis zur Sekunde vorgegeben und berücksichtigt. D.h. die Zeit der Öffnung jeder Bar, für die der Wert zurückgegeben wird (Volumen, Spread, Wert im Indikatorpuffer, Preis Open, High, Low, Close oder Zeit der Öffnung Time), ist immer gleich oder kleiner als das angegebene Datum.

Bei der Anforderung der Daten im vorgegebenen Datenbereich kehren nur die Daten zurück, die in das angeforderte Intervall kommen, dabei wird das Intervall mit Genauigkeit bis zur Sekunde vorgegeben und berücksichtigt. D.h. die Zeit der Öffnung jeder Bar, für die der Wert zurückgegeben wird (Volumen, Spread, Wert im Indikatorpuffer, Preis Open, High, Low, Close oder Zeit der Öffnung Time) befindet sich immer im angeforderten Intervall.

So wenn der laufende Wochentag Samstag ist, kehrt die Funktion beim Versuch, Daten in dem wöchentlichen Timeframe mit Andeutung *start\_time=Letzter\_Dienstag* und *stop\_time=Letzter\_Freitag* zu kopieren, 0 zurück, denn die Eröffnungszeit im wöchentlichen Timeframe ist immer der Sonntag, aber keine angegebene Bar gerät in den angegebenen Bereich.

Wenn man muss, den ersten Wert zu erhalten, der der laufenden nicht beendeten Bar entspricht, kann man die erste Aufrufform mit der Andeutung *start\_pos=0* und *count=1* verwenden.

#### Beispiel:

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots 1
//---- plot Spread
#property indicator_label1 "Spread"
#property indicator_type1 DRAW_HISTOGRAM
#property indicator_color1 clrRed
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//--- input parameters
input int bars=3000;
//--- indicator buffers
double SpreadBuffer[];
//+-----+
//| Custom indicator initialization function |
//+-----+
void OnInit()
{
//--- indicator buffers mapping
SetIndexBuffer(0,SpreadBuffer,INDICATOR_DATA);
IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
;}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
```

```

        const double &high[],
        const double &low[],
        const double &close[],
        const long &tick_volume[],
        const long &volume[],
        const int &spread[])

{
//---
    if (prev_calculated==0)
    {
        int spread_int[];
        ArraySetAsSeries(spread_int,true);
        int spreads=CopySpread(Symbol(),0,0,bars,spread_int);
        Print("Wir haben die folgende Anzahl der historischen Werte des Spreads bekommen");
        for (int i=0;i<spreads;i++)
        {
            SpreadBuffer[rates_total-1-i]=spread_int[i];
            if(i<=30) Print("spread["+i+"] = ",spread_int[i]);
        }
    }
    else
    {
        double Ask,Bid;
        Ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
        Bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
        Comment("Ask = ",Ask," Bid = ",Bid);
        SpreadBuffer[rates_total-1]=(Ask-Bid)/Point();
    }
//--- return value of prev_calculated for next call
    return(rates_total);
}

```

Sehen Sie ein umfassenderes Beispiel von Anfrage der historischen Daten in [Methode der Objektbindung](#). Der Skript zeigt, wie Daten von Indikator [iFractals](#) an den letztem 1000 Balken zu erhalten und wie dann die letzten 10 Fraktale nach oben und 10 Fraktale nach unten auf dem Chart anzuzeigen. Sie können eine ähnliche Technik für alle Indikatoren, die fehlende Werte haben und typischerweise unter Verwendung der folgenden [Stil](#) hergestellt werden, verwenden:

- [DRAW\\_SECTION](#),
- [DRAW\\_ARROW](#),
- [DRAW\\_ZIGZAG](#),
- [DRAW\\_COLOR\\_SECTION](#),
- [DRAW\\_COLOR\\_ARROW](#),
- [DRAW\\_COLOR\\_ZIGZAG](#).

# CopyTicks

Die Funktion erhält für die aktuelle Arbeitssitzung angesammelten Ticks ins Array `ticks_array`. Die Reihenfolge der Indizierung ist von der Vergangenheit in die Gegenwart, also der Tick mit Index 0 ist der älteste im Array.

```
int CopyTicks(
    string            symbol_name,           // Symbolname
    MqlTick&         ticks_array[],         // Array für Ticks
    uint              flags=COPY_TICKS_ALL,  // das Flag, das Typ der Ticks angibt
    ulong             from=0,               // das Datum ab dem Ticks angefordert werden
    uint              count=0               // Anzahl der Ticks, die Sie erhalten wollen
);
```

### Parameter

*symbol name*

[in] СИМВОЛ.

```
ticks_array
```

[out] Array vom Type `MqlTick` für Empfangen von Ticks.

*flags*

[in] Das Flag, das Typ der Ticks angibt. **COPY\_TICKS\_INFO** - nur Bid und Ask, **COPY\_TICKS\_TRADE** - nur Last und Volume, **COPY\_TICKS\_ALL** - alle Ticks.

*from*

[in] Das Datum, ab dem Ticks angefordert werden. In Millisekunden seit 01.01.1970.

*count*

[in] Anzahl der abgefragten Ticks. Wenn das Datum und die Anzahl nicht angegeben werden, werden alle verfügbaren Ticks, jedoch nicht mehr als 2000 Ticks, übertragen werden.

## Rückgabewert

Anzahl der kopierten Ticks oder -1 bei einem Fehler.

### Bemerkung

Funktion `OnTick()` handelt nicht alle Ticks, sondern unterrichtet sie dem Expert von der Veränderungen im Markt. Änderungen können wie ein Batch kommen: das Terminal kann gleichzeitig einige Ticks erreichen, aber die Funktion `onTick()` wird nur einmal aufgerufen, um den Experten auf dem neuesten Stand des Marktes zu informieren. Die Funktion `CopyTicks()` erlaubt alle empfangene Ticks abzufragen und zu analysieren.

Wenn die Parameter *from* und *count* nicht angegeben sind, dann alle verfügbare Ticks, aber nicht mehr als 2000, werden in *ticks\_array[]* angeschrieben werden. Typ von Ticks kann im Parameter *flags* angegeben werden.

### Beispiel:

```
//--- input parameters  
input int      ticks=10; // Anzahl der abgefragten Ticks  
//+-----
```

```

///| Expert tick function |
//+-----+
void OnTick()
{
//--- Array für Ticks
    MqlTick tick_array[];
//--- Ticks abfragen
    int copied=CopyTicks(_Symbol,tick_array,COPY_TICKS_ALL,0,ticks);
//--- Wenn die Ticks erhalten werden, zeigen wir die Werte von Bid und Ask auf dem Chart
    if(copied>0)
    {
        string comment="#   Time       Bid       Ask\r\n";
        //--- Inhalt der Kommentare generieren
        for(int i=0;i<copied;i++)
        {
            MqlTick tick=tick_array[i];
            string tick_string=StringFormat("%d: %s  %G  %G",
                                             i,
                                             TimeToString(tick.time,TIME_MINUTES|TIME_SECONDS),
                                             tick.bid,
                                             tick.ask);

            comment=comment+tick_string+"\r\n";
        }
        //--- Kommentare auf dem Chart anzeigen
        Comment(comment);
    }
    else // ein Fehler beim Erhalten von Ticks
    {
        Comment("Konnte nicht Ticks laden. GetLastError()=",GetLastError());
    }
}

```

Sehen Sie auch

[SymbolInfoTick](#), [Structure for Current Prices](#), [OnTick\(\)](#)



## Operationen mit Charts

Funktionen für die Arbeit mit Charts. Alle Operationen sind erst in Experten und Scripts verwendbar.

Funktionen, die Eigenschaften des Charts setzen, werden in der Tat verwendet, um Befehle zu ändern zu senden. Der erfolgreiche Erfüllung dieser Funktionen wird der Befehl in die allgemeine Warteschlange der Chartereignisse platziert. Der Chart wird im Zuge der Verarbeitung der Ereigniswarteschlange von diesem Chart geändert.

Aus diesem Grund sollte man nicht erwarten, dass der Chart sofort nach dem Aufruf dieser Funktionen visuell aktualisiert wird. Im Allgemeinen ist der Chart durch das Terminal automatisch nach Änderungs-Ereignisse aktualisiert - die Ankunft der neuen Quote, Änderung von Chartfenstergröße, etc.

Um Aussehen des Charts zu aktualisieren, verwenden Sie [ChartRedraw\(\)](#).

| Funktion                           | Massnahme                                                                                            |
|------------------------------------|------------------------------------------------------------------------------------------------------|
| <a href="#">ChartApplyTemplate</a> | Verwendet Schablone zum bezeichneten Chart der bezeichneten Datei                                    |
| <a href="#">ChartSaveTemplate</a>  | Speichert aktuelle Chart-Einstellungen in einer Schablone mit einem bestimmten Namen                 |
| <a href="#">ChartWindowFind</a>    | Bringt die Nummer des Subfensters, in dem Indikator war, zurück                                      |
| <a href="#">ChartTimePriceToXY</a> | Konvertiert die Koordinaten des Charts aus der Darstellung von Zeit/Geld in die X- und Y-Koordinaten |
| <a href="#">ChartXYToTimePrice</a> | Konvertiert die X- und Y-Koordinate des Charts in Zeit und Preis                                     |
| <a href="#">ChartOpen</a>          | Oeffnet einen neuen Chart mit dem bezeichneten Symbol und mit der bezeichneten Periode               |
| <a href="#">ChartClose</a>         | Schliesst den bezeichneten Chart ab                                                                  |
| <a href="#">ChartFirst</a>         | Bringt Identifikator des ersten Kundenterminals zurück                                               |
| <a href="#">ChartNext</a>          | Bringt Identifikator des Charts, das dem bezeichneten Chart folgt, zurück                            |
| <a href="#">ChartSymbol</a>        | Bringt den Symbolnamen des bezeichneten Chart zurück                                                 |
| <a href="#">ChartPeriod</a>        | Bringt den Periodenwert des bezeichneten Chart zurück                                                |
| <a href="#">ChartRedraw</a>        | Ruft Zwangsumzeichnen des bezeichneten Charts hervor                                                 |
| <a href="#">ChartSetDouble</a>     | Ordnet double-Typ Wert der entsprechenden Eigenschaft des bezeichneten Charts zu                     |

|                                             |                                                                                                                                    |
|---------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#"><u>ChartSetInteger</u></a>      | Ordnet Ganzzwert (datetime, int, color, bool oder char) der entsprechenden Eigenschaft des bezeichneten Charts zu                  |
| <a href="#"><u>ChartSetString</u></a>       | Ordnet string-Typ Wert der entsprechenden Eigenschaft des bezeichneten Charts zu                                                   |
| <a href="#"><u>ChartGetDouble</u></a>       | Bringt Wert der entsprechenden Eigenschaft des bezeichneten Charts zurück                                                          |
| <a href="#"><u>ChartGetInteger</u></a>      | Bringt Ganzzwert der entsprechenden Eigenschaft des bezeichneten Charts zurück                                                     |
| <a href="#"><u>ChartGetString</u></a>       | Bringt Zeilenwert der entsprechenden Eigenschaft des bezeichneten Charts zurück                                                    |
| <a href="#"><u>ChartNavigate</u></a>        | Verschiebt den angegebenen Chart um die angegebene Anzahl der Bars in Bezug auf die angegebene Position des Charts                 |
| <a href="#"><u>ChartID</u></a>              | Bringt Indikator des laufenden Charts zurück                                                                                       |
| <a href="#"><u>ChartIndicatorAdd</u></a>    | Setzt Indikator mit dem angegebenen handle ins angegebene Fenster der graphischen Darstellung zu                                   |
| <a href="#"><u>ChartIndicatorDelete</u></a> | Löscht Indikator mit angegebenem Namen aus dem angegebenen Chartfenster                                                            |
| <a href="#"><u>ChartIndicatorGet</u></a>    | Gibt das Handle des Indikators mit dem angegebenen Kurznamen im angegebenen Chart-Fenster zurück                                   |
| <a href="#"><u>ChartIndicatorName</u></a>   | Bringt Kurzname des Indikators beim Nummer in der Indikatorenliste des Chartfensters zurück                                        |
| <a href="#"><u>ChartIndicatorsTotal</u></a> | Bringt die Anzahl aller Indikatoren am angegebenen Chartfenster                                                                    |
| <a href="#"><u>ChartWindowOnDropped</u></a> | Bringt Nummer des Subfensters zurück, wohin der vorliegende Expert, Script, Objekt oder Anzeiger mit der Maus versetzt wurden sind |
| <a href="#"><u>ChartPriceOnDropped</u></a>  | Bringt die Preiskoordinate des Punktes zurück, wohin dieser Script oder Expert mit der Maus versetzt wurde                         |
| <a href="#"><u>ChartTimeOnDropped</u></a>   | Bringt die Preiskoordinate des Punktes zurück, wohin dieser Expert oder Script mit der Maus versetzt wurde.                        |
| <a href="#"><u>ChartXOnDropped</u></a>      | Bringt X-Koordinate zurück, wohin dieser Expert oder Script mit der Maus versetzt wurde                                            |
| <a href="#"><u>ChartYOnDropped</u></a>      | Bringt Y-Koordinate zurück, wohin dieser Expert oder Script mit der Maus versetzt wurde                                            |
| <a href="#"><u>ChartSetSymbolPeriod</u></a> | Verändert die Werte des Symbols und der                                                                                            |

|                        |                                                                                               |
|------------------------|-----------------------------------------------------------------------------------------------|
|                        | Periode des angegebenen Charts                                                                |
| <u>ChartScreenShot</u> | Screenshot des angegebenen Charts in Format GIF, PNG oder BMP je nach angegebenen Erweiterung |

## ChartApplyTemplate

Wendet zum Chart die angegebene Schablone zu. Der Befehl wird in die Warteschlange der Charnachrichten gestellt und wird nach der Prozessierung aller vorherigen Befehle abgearbeitet werden.

```
bool ChartApplyTemplate(  
    long      chart_id,      // Identifikator des Charts  
    const string filename    // Dateiname mit der Schablone  
);
```

### Parameter

*chart\_id*

[in] Identifikator des Charts. 0 bedeutet den laufenden Chart.

*filename*

[in] Dateiname mit dem Symbol.

### Rückgabewert

Gibt true zurück, wenn der Befehl in die Warteschlange des Charts gestellt wurde, andernfalls gibt false zurück. Um die Information über den [Fehler](#) zu erhalten, muss man die Funktion [GetLastError\(\)](#) aufrufen.

### Bemerkung

Wenn durch diese Funktion eine neue Schablone in den Chart geladen wird, zu der die Schablone angehängt ist, wird Expert ausgeladen und seine Arbeit nicht fortsetzen kann.

В целях безопасности право на торговлю при применении шаблона к графику может ограничиваться:

**Права на торговлю не могут быть повышены при запуске советника путем применения шаблона с помощью функции ChartApplyTemplate().**

Если у mql5-программы, которая вызывает функцию ChartApplyTemplate(), отсутствуют права на торговлю, то эксперт, загруженный при помощи шаблона, также не будет иметь прав на торговлю вне зависимости от настроек шаблона.

Если у mql5-программы, которая вызывает функцию ChartApplyTemplate(), есть права на торговлю, а в настройках шаблона права отсутствуют, то советник, загруженный при помощи шаблона, не будет иметь прав на торговлю.

## Verwenden von Schablonen

Mit Hilfe der Mittel der MQL5-Sprache können Sie viele Eigenschaften des Charts definieren, einschließlich der Farben mit der [ChartSetInteger\(\)](#)-Funktion:

- Hintergrundfarbe des Charts;
- Farbe der Achsen, Skalen und OHLC-Reihen;
- Grid-Farbe;
- Farbe der Volumen und Position offen Ebenen;

- Farbe des up-Bars, Schatten und Rand eines bullish Candlestick;
- Farbe des unten-Bars, Schatten und Rand eines bullish Candlestick;
- Farbe der Chartlinien und Doji Candlesticks;
- Farbe der bullischen Candlestick Körper;
- Farbe der bearishen Candlestick Körper;
- Farbe der Bid-Preis Linie;
- Farbe der Ask-Preis Linie;
- Farbe der Linie der letzten Transaktion (Last);
- Farbe der Stop-Order Ebenen (Stop Loss und Take Profit).

Außerdem kann es mehrere [grafische Objekte](#) und [Indikatoren](#) in einem Chart werden. Sie können ein Chart mit allen notwendigen Indikatoren einmal abstimmen und speichern es als Schablone. Eine solche Schablone kann auf jeden Chart aufgebracht werden.

Die Funktion [ChartApplyTemplate\(\)](#) ist für die Verwendung einer zuvor gespeicherten Schablone bestimmt, und es kann in jeder MQL5 Programm verwendet werden. Der Pfad zu der Datei, in der die Schablone liegt, wird als zweiter Parameter zu ChartApplyTemplate() übergeben. Die Schablone-Datei wird nach folgenden Regeln gesucht:

- wenn der Backslash "\" Separator (wie geschrieben "\\") wird am Anfang des Pfads platziert, wird die Schablone relativ zum Pfad \_Terminaldaten\_Verzeichnis\MQL5 gesucht,
- wenn es kein Backslash gibt, wird die Schablone relativ zur ausführbaren EX5-Datei, in denen ChartApplyTemplate() aufgerufen wird, gesucht;
- wenn die Schablone nicht in den ersten beiden Varianten gefunden ist, wird die Suche in dem Ordner Terminal\_Verzeichnis\Profiles\Templates\ durchgeführt.

Hier Terminal\_Verzeichnis ist der Ordner, aus dem MetaTrader 5 Client Terminal läuft, und Terminaldaten\_Verzeichnis ist der Ordner, in denen editierbare Dateien gespeichert sind, sein Lage ist auf dem Betriebssystem, den Benutzernamen und Computer-Security-Einstellungen abhängig. Normalerweise werden sie verschiedenen Ordnern, sondern in einigen Fällen können sie zusammenfallen.

Der Speicherort der Ordner Terminaldaten\_Verzeichnis und Terminal\_Verzeichnis kann mit der Funktion [TerminalInfoString\(\)](#) erfährt werden.

```
//--- Verzeichnis, aus dem das Terminal gestartet war
string terminal_path=TerminalInfoString(TERMINAL_PATH);
Print("Terminal_Verzeichnis:",terminal_path);
//--- Terminaldaten_Verzeichnis, in dem der MQL5-Ordner mit EAs und Indikatoren liegt
string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
Print("Terminaldaten_Verzeichnis:",terminal_data_path);
```

Beispiele:

```
//--- Suche nach einer Schablone in Terminaldaten_Verzeichnis\MQL5\
ChartApplyTemplate(0,"\\first_template.tpl")
//--- Suche nach einer Schablone in EX5-Datei_Verzeichnis\, dann in Terminaldaten_Verzeichnis\
ChartApplyTemplate(0,"second_template.tpl")
//--- Suche in EX5-Datei_Verzeichnis\My_templates\, dann in Terminaldaten_Verzeichnis\
ChartApplyTemplate(0,"My_templates\\third_template.tpl")
```

Schablonen sind nicht Ressourcen, können sie nicht in eine ausführbare EX5-Datei einbezogen werden.

**Beispiel:**

```
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//--- пример применения шаблона, расположенного в каталоге \MQL5\Files
    if(FileExists("my_template.tpl"))
    {
        Print("Шаблон my_template.tpl найден в каталоге \Files");
        //--- применим найденный шаблон
        if(CharApplyTemplate(0,"\\Files\\my_template.tpl"))
        {
            Print("Применили успешно шаблон 'my_template.tpl'");
            //--- принудительно перерисуем график для быстрого показа изменений
            ChartRedraw();
        }
    }
    else
        Print("Не удалось применить шаблон 'my_template.tpl', ошибка ",GetLastError());
}
else
{
    Print("Файл 'my_template.tpl' не найден в папке "
        +TerminalInfoString(TERMINAL_PATH)+"\\MQL5\\Files");
}
}
```

**Sehen Sie auch**[Ressourcen](#)

## ChartSaveTemplate

Speichert die aktuelle Chart-Einstellungen in der Schablone mit dem angegebenen Namen.

```
bool ChartSaveTemplate(
    long      chart_id,      // Chart ID
    const string filename    // Dateiname für die Schablone
);
```

### Parameter

*chart\_id*

[in] Die ID des Charts. 0 bedeutet den aktuellen Chart.

*filename*

[in] Dateiname um die Schablone zu speichern. Die Erweiterung ".tpl" wird an den Dateinamen automatisch hinzugefügt, so müssen Sie nicht es angeben. Die Schablone ist im Ordner **Terminal\_Ordner\Profiles\Templates\** gespeichert und kann auch für manuelle Anwendung im Terminal verwendet werden. Wenn eine Schablone mit diesem Namen bereits existiert, wird ihr Inhalt überschrieben werden.

### Rückgabewert

Bei der erfolgreichen Anwendung der Schablone gibt die Funktion true zurück, anderenfalls gibt sie false zurück. Um die Information über [Fehler](#) zu erhalten, muss man die Funktion [GetLastError\(\)](#) aufrufen.

### Bemerkung

In einer Schablone können Sie die Einstellungen des Charts mit allen darauf aufgesetzten Indikatoren und Grafiken speichern, damit sie auf ein anderer Chart zu anwenden.

### Beispiel:

```
//+-----+
//|                                     Test_ChartSaveTemplate.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property script_show_inputs

/-- input parameters
input string      symbol="GBPUSD"; // Symbol des neuen Charts
input ENUM_TIMEFRAMES period=PERIOD_H3; // Timeframe des neuen Charts
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    /-- Zuerst Indikatoren auf den Chart aufsetzen
```

```

int handle;
//--- Den Indikator zu nutzen vorbereiten
if(!PrepareZigzag(NULL,0,handle)) return; // Es hat nicht geklappt
//--- Den Indikator auf den aktuellen Chart aber einem separaten Fenster aufsetzen
if(!ChartIndicatorAdd(0,1,handle))
{
    PrintFormat("Fehler beim Aufsetzung von Indikator mit Handle=%d auf Chart %s/%s.",
        _Symbol,
        EnumToString(_Period),
        handle,
        GetLastError());
    //--- Das Programm vorzeitig beenden
    return;
};
//--- Den Chart aktualisieren, im den Indikator zu sehen
ChartRedraw();
//--- Hier finden wir zwei letzten Zick-Zack-Frakturen
double two_values[];
datetime two_times[];
if(!GetLastTwoFractures(two_values,two_times,handle))
{
    PrintFormat("Fehler beim Such der zwei letzten Zick-Zack-Frakturen im Indikator
    //--- Das Programm vorzeitig beenden
    return;
};
//--- Jetzt Standardabweichung Kanal aufsetzen
string channel="StdDeviation Channel";
if(!ObjectCreate(0,channel,OBJ_STDDEVCHANNEL,0,two_times[1],0))
{
    PrintFormat("Objekt %s konnte nicht erstellt werden. Fehlercode %d",
        EnumToString(OBJ_STDDEVCHANNEL),GetLastError());
    return;
};
else
{
    //--- Der Kanal erstellt wurde, definieren wir den zweiten Punkt
    ObjectSetInteger(0,channel,OBJPROP_TIME,1,two_times[0]);
    //--- Tooltip für Kanal definieren
    ObjectSetString(0,channel,OBJPROP_TOOLTIP,"Demo from MQL5 Help");
    //--- Chart aktualisieren
    ChartRedraw();
};
//--- Dies in einer Schablone speichern
ChartSaveTemplate(0,"StdDevChannelOnZigzag");
//--- Ein neuer Chart öffnen und die Schablone darauf aufsetzen
long new_chart=ChartOpen(symbol,period);
//--- Tooltips für graphische Objekten entscheiden
ChartSetInteger(new_chart,CHART_SHOW_OBJECT_DESCR,true);
if(new_chart!=0)

```



```

    {
        //--- Die neue Schablone auf den Chart aufsetzen
        ChartApplyTemplate(new_chart,"StdDevChannelOnZigzag");
    };
    Sleep(10000);
};

//+-----+
//| Erzeugt Handle auf Zick-Zack, sorgt für Bereitschaft seinen Daten|
//+-----+
bool PrepareZigzag(string sym,ENUM_TIMEFRAMES tf,int &h)
{
    ResetLastError();
    //--- Indikator Zigzag muss in Terminal_Angaben_Ordner\MQL5\Examples sein
    h=iCustom(sym,tf,"Examples\\Zigzag");
    if(h==INVALID_HANDLE)
    {
        PrintFormat("%s: Handle von Zigzag konnte nicht erstellt werden. Fehlercode %d",
                    __FUNCTION__,GetLastError());
        return false;
    };
    //--- Bei der Erstellung eines Indikatorhandles, benötigt es Zeit um Werte zu berechnen
    int k=0; // Die Anzahl der Versuche für Warten der Berechnung des Indikators
    //--- Auf die Berechnung in Zyklus warten, eine Pause von 50 Millisekunden machen, wenn
    while(BarsCalculated(h)<=0)
    {
        k++;
        //--- Die Anzahl der Versuche anzeigen
        PrintFormat("%s: k=%d",__FUNCTION__,k);
        //--- 50 Millisekunden warten, während der Indikator berechnet wird
        Sleep(50);
        //--- Wenn mehr als 100 Versuche gemacht worden, dann ist etwas falsch
        if(k>100)
        {
            //--- Problem melden
            PrintFormat("Indikator konnte nicht in %d Versuche berechnet werden!");
            //--- Problem melden
            return false;
        };
    };
    //--- Alles ist fertig, der Indikator ist erstellt und die Werte sind berechnet
    return true;
};

//+-----+
//| Sucht für die letzten 2 Zick-Zack-Frakturen und setzt in Arrays |
//+-----+
bool GetLastTwoFractures(double &get_values[],datetime &get_times[],int handle)
{
    double values[]; // Array für die Werte der Zick-Zack
    datetime times[]; // Array, um Zeit zu bekommen

```

```

int size=100;           // Größe der Arrays
ResetLastError();
//--- Letzten 100 Werte kopieren
int copied=CopyBuffer(handle,0,0,size,values);
//--- Die Anzahl der kopierten Werte überprüfen
if(copied<100)
{
    PrintFormat("%s: %d Werte des Indikators mit Handle =%d konnte nicht kopiert werden",
        __FUNCTION__,size,handle,GetLastError());
    return false;
};
//--- Reihenfolge der Zugriff auf das Array als in einer Zeitreihe definieren
ArraySetAsSeries(values,true);
//--- Die Nummer der Bars mit Frakturen hier schreiben
int positions[];
//--- Dimensionen der Arrays definieren
ArrayResize(get_values,3); ArrayResize(get_times,3); ArrayResize(positions,3);
//--- Zähler
int i=0,k=0;
//--- Start der Suche nach Frakturen
while(i<100)
{
    double v=values[i];
    //--- NULL-Werte interessieren uns hier nicht
    if(v!=0.0)
    {
        //--- Merken wir uns die Nummer des Bars
        positions[k]=i;
        //--- Merken wir uns den Wert des Zick-Zacks auf die Fraktur
        get_values[k]=values[i];
        PrintFormat("%s: Zigzag[%d]=%G",__FUNCTION__,i,values[i]);
        //--- Erhöhung des Zählers
        k++;
        //--- Wenn zwei Frakturen sind gefunden, brechen den Zyklus
        if(k>2) break;
    };
    i++;
};
//--- Reihenfolge der Zugriff auf das Array als in einer Zeitreihe definieren
ArraySetAsSeries(times,true); ArraySetAsSeries(get_times,true);
if(CopyTime(_Symbol,_Period,0,size,times)<=0)
{
    PrintFormat("%s: %d Werte aus CopyTime() konnten nicht kopiert werden. Fehlercode: %d",
        __FUNCTION__,size,GetLastError());
    return false;
};
//--- Finden die Öffnungszeit von Bars, an denen die 2 Frakturen sind
get_times[0]=times[positions[1]]; // Der vorletzte Wert ist als erster Fraktur gesch
get_times[1]=times[positions[2]]; // Dritter von Ende Wert ist als zweiter Fraktur g

```

```
PrintFormat("%s: Erst=%s, Zweit=%s", __FUNCTION__, TimeToString(get_times[1]), TimeTo
//--- Fertig
return true;
; }
```

Sehen Sie auch

[ChartApplyTemplate\(\)](#), [Ressourcen](#)

## ChartWindowFind

Gibt die Nummer des Subfensters zurück, in dem sich Indikator befindet. Es gibt zwei Varianten der Funktion.

1. Funktion sucht auf dem angegebenen Chart das Subfenster mit dem angegebenen "kurzen Namen" des Indikatoren (kurzer Name wird im oberen linken Teil des Subfensters ausgegeben) und im Erfolgsfall gibt die Nummer des Subfensters zurück.

```
int ChartWindowFind(  
    long      chart_id,           // Identifikator des Charts  
    string    indicator_shortcode // kurzer Name des Indikatoren, s. INDICATOR\_SHORTNAME)
```

2. Funktion muss vom Benutzerindikator aufgerufen werden und sie gibt die Nummer des Subfensters, wo Indikator arbeitet, zurück.

```
int ChartWindowFind();
```

### Parameter

*chart\_id*

[in] Identifikator des Charts. 0 bedeutet den laufenden Chart.

*indicator\_shortcode*

[in] Kurzer Name des Anzeigers.

### Rückgabewert

Nummer des Subfensters im Erfolgsfall. Null bedeutet Hauptfenster des Charts. Im Fall des Misserfolges gibt -1 zurück.

### Bemerkung

Wenn die zweite Variante der Funktion (ohne Parameter) aus Script oder Experten aufgerufen wird, gibt sie -1 zurück.

Mann sollte nicht Indikators Kurzname mit dem Dateinamen der beim Erstellen des Indikators mit den Funktionen [iCustom\(\)](#) und [IndicatorCreate\(\)](#) gezeigt wird verwechseln. Wenn der Kurzname des Indikators nicht explizit festgelegt wird, dann bei der Kompilation wird den Namen der Quelldatei des Indikators angegeben.

Sie sollen richtig einen kurzen Namen des Indikators formen, der in der Eigenschaft [INDICATOR\\_SHORTNAME](#) mit Hilfe der Funktion [IndicatorSetString\(\)](#) geschrieben wird. Wir empfehlen, dass der Kurzname die Werte der Eingabeparameter des Indikators enthält, da der Indikator, der aus dem Chart gelöscht wird, in der Funktion [ChartIndicatorDelete\(\)](#) beim Kurzname identifiziert wird.

### Beispiel:

```

#property script_show_inputs
//--- input parameters
input string   shortName="MACD(12,26,9)";
//+-----+
//| Gibt Nummer des Charts mit dem angegebenen Indikator zurück |
//+-----+
int GetIndicatorSubWindowNumber(long chartID=0,string short_name="")
{
    int window=-1;
//---
    if((ENUM_PROGRAM_TYPE)MQL5InfoInteger(MQL5_PROGRAM_TYPE)==PROGRAM_INDICATOR)
    {
        //--- Funktion wird vom Indikator aufgerufen, der Name ist nicht erforderlich.
        window=ChartWindowFind();
    }
    else
    {
        //--- Funktion wird vom Experten oder Script aufgerufen
        window=ChartWindowFind(0,short_name);
        if(window==-1) Print(__FUNCTION__+"() : Error = ",GetLastError());
    }
//---
    return(window);
}
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//---
    int window=GetIndicatorSubWindowNumber(0,shortName);
    if(window!=-1)
        Print("Anzeiger "+shortName+" befindet sich im Fenster #"+(string)window);
    else
        Print("Anzeiger "+shortName+" ist nicht gefunden. window = "+(string)window);
}

```

Sehen Sie auch

[ObjectCreate\(\)](#), [ObjectFind\(\)](#)

## ChartTimePriceToXY

Konvertiert die Koordinaten des Charts aus der Darstellung von Zeit/Geld in die X- und Y- Koordinaten.

```
bool ChartTimePriceToXY(  
    long      chart_id,    // ID des Charts  
    int       sub_window,  // Nummer des Unterfensters  
    datetime  time,        // Zeit auf dem Chart  
    double    price,       // Preis auf dem Chart  
    int&      x,           // X-Koordinate für die Zeit auf dem Chart  
    int&      y            // Y-Koordinate für den Preis auf dem Chart  
);
```

### Parameter

*chart\_id*

[in] ID des Charts. 0 bedeutet den aktuellen Chart.

*sub\_window*

[in] Nummer des Unterfensters. 0 bedeutet das Hauptfenster des Charts.

*time*

[in] Der Wert der Zeit auf dem Chart, für den ein Wert in Pixels auf der X-Achse erhalten wird. Der Ursprung ist in der oberen linken Ecke des Hauptfensters des Charts.

*price*

[in] Der Wert des Preises auf dem Chart, für den ein Wert in Pixels auf der Y-Achse erhalten wird. Der Ursprung ist in der oberen linken Ecke des Hauptfensters des Charts.

*x*

[out] Die Variable, in die Konvertierung von Zeit in X-Koordinate erhalten wird.

*y*

[out] Die Variable, in die Konvertierung von Preis in Y-Koordinate erhalten wird.

### Rückgabewert

Gibt bei Erfolg true zurück, ansonsten false. Um Informationen über den [Fehler](#) zu erhalten, rufen Sie die Funktion [GetLastError\(\)](#).

### Sehen Sie auch

[ChartXYToTimePrice\(\)](#)

## ChartXYToTimePrice

Konvertiert die X- und Y-Koordinate des Charts in Zeit und Preis.

```
bool ChartXYToTimePrice(  
    long      chart_id,    // ID des Charts  
    int       x,           // X-Koordinate auf dem Chart  
    int       y,           // Y-Koordinate auf dem Chart  
    int&      sub_window,  // Nummer des Unterfensters  
    datetime& time,        // Zeit auf dem Chart  
    double&   price        // Preis auf dem Chart  
);
```

### Parameter

*chart\_id*

[in] ID des Charts. 0 bedeutet den aktuellen Chart.

*x*

[in] X-Koordinate.

*y*

[in] Y-Koordinate.

*sub\_window*

[out] Die Variable, in die die Nummer des Chartunterfenster geschrieben wird. 0 bedeutet das Hauptfenster des Charts.

*time*

[out] Der Wert der Zeit auf dem Chart, für den ein Wert in Pixels auf der X-Achse erhalten wird. Der Ursprung ist in der oberen linken Ecke des Hauptfenster des Charts.

*price*

[out] Der Wert des Preises auf dem Chart, für den ein Wert in Pixels auf der Y-Achse erhalten wird. Der Ursprung ist in der oberen linken Ecke des Hauptfenster des Charts.

### Rückgabewert

Gibt bei Erfolg true zurück, ansonsten false. Um Informationen über den [Fehler](#) zu erhalten, rufen Sie die Funktion [GetLastError\(\)](#).

### Beispiel:

```

//+-----+
//| ChartEvent function |
//+-----+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
{
    //--- Zeigen die Parameter des Ereignisses auf dem Chart
    Comment(__FUNCTION__, ": id=", id, " lparam=", lparam, " dparam=", dparam, " sparam=", sparam);
    //--- wenn es ein Ereignis von Mausklick auf dem Chart ist
    if(id==CHARTEVENT_CLICK)
    {
        //--- Vorbereitung der Variablen
        int x = (int)lparam;
        int y = (int)dparam;
        datetime dt = 0;
        double price = 0;
        int window = 0;
        //--- Konvertieren wir X und Y in Zeit/Preis
        if(ChartXYToTimePrice(0, x, y, window, dt, price))
        {
            PrintFormat("Window=%d X=%d Y=%d => Time=%s Price=%G", window, x, y, TimeToString(dt), price);
            //--- inverse Umwandlung: (X,Y) => (Time,Price)
            if(CharTimePriceToXY(0, window, dt, price, x, y))
                PrintFormat("Time=%s Price=%G => X=%d Y=%d", TimeToString(dt), price, x, y);
            else
                Print("ChartTimePriceToXY return error code: ", GetLastError());
            //--- delete lines
            ObjectDelete(0, "V Line");
            ObjectDelete(0, "H Line");
            //--- create horizontal and vertical lines of the crosshair
            ObjectCreate(0, "H Line", OBJ_HLINE, window, dt, price);
            ObjectCreate(0, "V Line", OBJ_VLINE, window, dt, price);
            ChartRedraw(0);
        }
        else
            Print("ChartXYToTimePrice return error code: ", GetLastError());
        Print("+-----+");
    }
}

```

Sehen Sie auch

[ChartTimePriceToXY\(\)](#)



## ChartOpen

Eroffnet einen neuen Chart mit dem angegebenen Symbol und Periode.

```
long ChartOpen(  
    string      symbol,      // Symbolname  
    ENUM_TIMEFRAMES period   // Periode  
);
```

### Parameter

*symbol*

[in] Symbol des Charts. [NULL](#) bedeutet Symbol des laufenden Charts (zu dem Expert angehängt wird).

*period*

[in] Periode des Charts (Timeframe). Kann einen der Aufzählungswerte [ENUM\\_TIMEFRAMES](#) annehmen. 0 bedeutet Periode des laufenden Charts.

### Rückgabewert

Bei der erfolgreichen Eroffnung des Charts gibt die Funktion Identifikator des Charts zurück. Anderenfalls gibt sie 0 zurück.

### Bemerkung

Maximal mögliche Zahl der gleichzeitig geöffneten Charts im Terminal kann nicht mehr als der Wert [CHARTS\\_MAX](#) sein.

## ChartFirst

Gibt Identifikator des ersten Charts des Kundenterminals zurück.

```
long ChartFirst();
```

### Rückgabewert

Identifikator des Charts.

## ChartNext

Funktion gibt Identifikator des Charts zurück, der nach dem angegebenen folgt.

```
long ChartNext(  
    long chart_id    // Identifikator des Charts  
);
```

### Parameter

*chart\_id*

[in] Identifikator des Charts. 0 bedeutet nicht den laufenden Chart. 0 bedeutet "identifikator des ersten Charts zurückgeben".

### Rückgabewert

Identifikator des Charts. Wenn die Liste der Charts zu Ende ist, gibt die Funktion -1 zurück.

### Beispiel:

```
//--- Variablen für Identifikatoren der Charts  
long currChart,prevChart=ChartFirst();  
int i=0,limit=100;  
Print("ChartFirst = ",ChartSymbol(prevChart)," ID = ",prevChart);  
while(i<limit)// sie haben bestimmt mehr als 100 geoeffnete Dateien  
{  
    currChart=ChartNext(prevChart); // bekommen einen neuen Chart mittels des vorherigen  
    if(currChart<0) break;           // haben das Ende der Chartliste erreicht  
    Print(i,ChartSymbol(currChart)," ID =",currChart);  
    prevChart=currChart;//speichern wir Identifikator des laufenden Charts für ChartNext  
    i++;// nicht vergessen, den Counter zu steigern  
};
```

## ChartClose

Schliesst den angegebenen Chart.

```
bool ChartClose(  
    long chart_id=0    // Identifikator des Charts  
);
```

### Parameter

*chart\_id=0*

[in] Identifikator des Charts. 0 bedeutet den laufenden Chart.

### Rückgabewert

Gibt true im Erfolgsfall zurück, anderenfalls false.

## ChartSymbol

Gibt den Symbolnamen des angegebenen Charts zurück.

```
string ChartSymbol(  
    long chart_id=0    // Identifikator des Charts  
);
```

### Parameter

*chart\_id=0*

[in] Identifikator des Charts. 0 bedeutet den laufenden Chart.

### Rückgabewert

Wenn der Chart nicht existiert, kehrt die Leerzeile zurück.

Sehen Sie auch

[ChartSetSymbolPeriod](#)

## ChartPeriod

Gibt den Wert Periode des angegebenen Charts zurück.

```
ENUM_TIMEFRAMES ChartPeriod(  
    long chart_id=0    // Identifikator des Charts  
);
```

### Parameter

*chart\_id=0*

[in] Identifikator des Charts. 0 bedeutet den laufenden Chart.

### Rückgabewert

Wert des Typs ENUM\_TIMEFRAMES. Wenn der Chart nicht existiert, kehrt 0 zurück.

## ChartRedraw

Ruft Zwangszurückzeichnung des angegebenen Charts.

```
void ChartRedraw(  
    long chart_id=0    // Identifikator des Charts
```

### Parameter

*chart\_id=0*

[in] Identifikator des Charts. 0 bedeutet den laufenden Chart.

### Bemerkung

Gewöhnlich wird nach der Veränderung der [Objekteigenschaften](#) verwendet.

Sehen Sie auch

[Graphische Objekte](#)

## ChartSetDouble

Gibt den Wert der entsprechenden Eigenschaft des angegebenen Charts vor. Eigenschaft des Charts muss des Typs [double](#) sein. Der Befehl wird in die Warteschlange der Chartnachrichten gestellt und wird nach der Prozessierung aller vorherigen Befehle abgearbeitet werden.

```
bool ChartSetDouble(  
    long    chart_id,      // Identifikator des Charts  
    int     prop_id,       // Identifikator der Eigenschaft  
    double  value          // Wert  
);
```

### Parameter

*chart\_id*

[in] Identifikator des Charts. 0 bedeutet den laufenden Chart.

*prop\_id*

[in] Identifikator der Eigenschaft des Charts. Wert kann einer der Aufzählungswerte [ENUM\\_CHART\\_PROPERTY\\_DOUBLE](#) sein (ausser read-only Eigenschaften).

*value*

[in] Wert der Eigenschaft.

### Rückgabewert

Gibt true zurück, wenn der Befehl in die Warteschlange des Charts gestellt wurde, andernfalls gibt false zurück. Um die Information über den [Fehler](#) zu erhalten, muss man die Funktion [GetLastError\(\)](#) aufrufen.



## ChartSetInteger

Gibt den Wert der entsprechenden Eigenschaft des angegebenen Charts vor. Eigenschaft des Charts muss der Typen [datetime](#), [int](#), [color](#), [bool](#) oder [char](#) sein. Der Befehl wird in die Warteschlange der Charnachrichten gestellt und wird nach der Prozessierung aller vorherigen Befehle abgearbeitet werden.

```
bool ChartSetInteger(  
    long   chart_id,      // Identifikator des Charts  
    int    prop_id,       // Identifikator der Eigenschaft  
    long   value          // Wert  
);
```

### Parameter

*chart\_id*

[in] Identifikator des Charts. 0 bedeutet den laufenden Chart.

*prop\_id*

[in] Identifikator der Eigenschaften des Charts. Wert kann einer der Aufzählungswerte [ENUM\\_CHART\\_PROPERTY\\_INTEGER](#) (ausser read-only Eigenschaften) sein.

*value*

[in] Wert der Eigenschaft.

### Rückgabewert

Gibt true zurück, wenn der Befehl in die Warteschlange des Charts gestellt wurde, andernfalls gibt false zurück. Um die Information über den [Fehler](#) zu erhalten, muss man die Funktion [GetLastError\(\)](#) aufrufen.

## ChartSetString

Gibt den Wert der entsprechenden Eigenschaft des angegebenen Charts vor. Eigenschaft des Charts muss des Typs string sein. Der Befehl wird in die Warteschlange der Chartnachrichten gestellt und wird nach der Prozessierung aller vorherigen Befehle abgearbeitet werden.

```
bool ChartSetString(
    long  chart_id,      // Identifikator des Charts
    int   prop_id,      // Identifikator der Eigenschaft
    string str_value     // Wert
);
```

### Parameter

*chart\_id*

[in] Identifikator des Charts. 0 bedeutet den laufenden Chart.

*prop\_id*

[in] Identifikator der Eigenschaften des Charts. Wert kann einer der Aufzählungswerte [ENUM\\_CHART\\_PROPERTY\\_STRING](#) (ausser read-only Eigenschaften) sein.

*str\_value*

[in] Zeile für Einstellung der Eigenschaft. Laenge der Zeile kann nicht mehr als 2045 Symbole sein (überschüssige Symbole werden entfernt werden).

### Rückgabewert

Gibt true zurück, wenn der Befehl in die Warteschlange des Charts gestellt wurde, andernfalls gibt false zurück. Um die Information über den [Fehler](#) zu erhalten, muss man die Funktion [GetLastError\(\)](#) aufrufen.

### Bemerkung

Funktion ChartSetString kann für Ausgabe der Kommentare auf den Chart statt der Funktion [Comment](#) verwendet werden.

### Beispiel:

```
void OnTick()
{
    //---
    double Ask,Bid;
    int Spread;
    Ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
    Bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
    Spread=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD);
    string comment=StringFormat("Ausgabe der Preise:\nAsk = %G\nBid = %G\nSpread = %d",
                                Ask,Bid,Spread);
    ChartSetString(0,CHART_COMMENT,comment);
};
```

### Sehen Sie auch

[Comment](#), [ChartGetString](#)

## ChartGetDouble

Gibt den Wert der entsprechenden Eigenschaft des angegebenen Charts zurück. Eigenschaft des Charts muss des Typs double sein. Es gibt 2 Varianten der Funktion.

1. Gibt den Wert der Eigenschaft direkt zurück.

```
double ChartGetDouble(
    long  chart_id,          // Identifikator des Charts
    int   prop_id,          // Identifikator der Eigenschaft
    int   sub_window=0      // Nummer des Subfensters wenn erforderlich
);
```

2. Gibt true oder false zurück abhängig von der erfolgreichen Durchführung der Funktion. Im Erfolgsfall wird der Wert der Eigenschaft in die Empfangsvariable gestellt, die durch Referenz vom letzten Parameter übertragen wird.

```
bool ChartGetDouble(
    long  chart_id,          // Identifikator des Charts
    int   prop_id,          // Identifikator der Eigenschaft
    int   sub_window,       // Nummer des Subfensters
    double& double_var      // hier erhalten wir Wert der Eigenschaft
);
```

### Parameter

*chart\_id*

[in] Identifikator des Charts. 0 bedeutet den laufenden Chart.

*prop\_id*

[in] Identifikator der Eigenschaft des Charts. Wert kann einer der Aufzählungswerte [ENUM\\_CHART\\_PROPERTY\\_DOUBLE](#) sein.

*sub\_window*

[in] Nummer des Subfensters des Charts. Für die erste Variante ist der Default-Wert 0 (Hauptfenster des Charts). Die meisten Eigenschaften erfordern die Nummer des Subfensters nicht.

[out] Variable des Typs double, die den Wert der angeforderten Eigenschaft annimmt.

### Rückgabewert

Wert des Typs double.

für die zweite Variante des Aufrufs gibt true zurück, wenn diese Eigenschaft zugaenglich ist und in der Variable double\_var aufbewahren wird, anderenfalls gibt false zurück. Für die zusätzliche Information über den [Fehler](#), rufen Sie die Funktion [GetLastError\(\)](#) auf.

### Beispiel:

```
void OnStart()
{
    double priceMin=ChartGetDouble(0,CHART_PRICE_MIN,0);
    double priceMax=ChartGetDouble(0,CHART_PRICE_MAX,0);
}
```

```
Print("CHART_PRICE_MIN = ",priceMin);  
Print("CHART_PRICE_MAX = ",priceMax);  
; }
```

## ChartGetInteger

Gibt den Wert der entsprechenden Eigenschaft des angegebenen Charts zurück. Die Eigenschaft des Charts muss der Typen [datetime](#), [int](#) oder [bool](#) sein. Es gibt 2 Varianten der Funktion.

1. Gibt den Wert der Eigenschaft direkt zurück.

```
long ChartGetInteger(
    long  chart_id,          // Identifikator des Charts
    int   prop_id,          // Identifikator der Eigenschaft
    int   sub_window=0      // Nummer des Subfensters wenn erforderlich
);
```

2. Gibt true oder false zurück, abhängig von der Erfolglosigkeit der Funktion. Im Erfolgsfall wird der Wert der Eigenschaft in die Empfangsvariable gestellt, die durch Referenz vom letzten Parameter übertragen wird.

```
bool ChartGetInteger(
    long  chart_id,          // Identifikator des Charts
    int   prop_id,          // Identifikator der Eigenschaft
    int   sub_window,       // Nummer des Subfensters
    long& long_var          // hier erhalten wir den Wert der Eigenschaft
);
```

### Parameter

*chart\_id*

[in] Identifikator des Charts. 0 bedeutet den laufenden Chart.

*prop\_id*

[in] Identifikator der Eigenschaft des Charts. Wert kann einer der Aufzählungswerte [ENUM\\_CHART\\_PROPERTY\\_INTEGER](#) sein.

*sub\_window*

[in] Nummer des Subfenster des Charts. Für den ersten Fall der Default-Wert ist 0 (Hauptfenster des Charts). Die meisten Eigenschaften ist die Nummer des Subfensters nicht erforderlich.

*long\_var*

[out] Variable des Typs long, die den Wert der angeforderten Eigenschaft annimmt.

### Rückgabewert

Wert des Typs long.

für die zweite Variante des Aufrufes gibt true zurück, wenn die Eigenschaft zugaenglich ist und der Wert in der Variable long\_var aufbewahren wird, anderenfalls gibt false zurück. Für die zusätzliche Information über das [Fehler](#), rufen Sie die Funktion [GetLastError\(\)](#) auf.

### Beispiel:

```
void OnStart()
{
    int height=ChartGetInteger(0, CHART_HEIGHT_IN_PIXELS, 0);
```

```
int width=ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0);  
Print("CHART_HEIGHT_IN_PIXELS = ",height," pixels");  
Print("CHART_WIDTH_IN_PIXELS = ",width," pixels");  
;}
```

## ChartGetString

Gibt den Wert der entsprechenden Eigenschaft des angegebenen Charts zurück. Die Eigenschaft des Charts muss des Typs string sein. Es gibt 2 Varianten der Funktion.

1. Gibt den Wert der Eigenschaft direkt zurück.

```
string ChartGetString(
    long   chart_id,           // Identifikator des Charts
    int    prop_id            // Identifikator der Eigenschaft
);
```

2. Gibt true oder false zurück, abhängig von der erfolgreichen Durchführung der Funktion. Im Erfolgsfall wird der Wert der Eigenschaft in die Empfangsvariable gestellt, die durch Referenz vom letzten Parameter übertragen wird.

```
bool ChartGetString(
    long   chart_id,           // Identifikator des Charts
    int    prop_id,           // Identifikator der Eigenschaft
    string& string_var        // hier erhalten wir den Wert der Eigenschaft
);
```

### Parameter

*chart\_id*

[in] Identifikator des Charts. 0 bedeutet den laufenden Chart.

*prop\_id*

[in] Identifikator der Eigenschaft des Charts. Wert kann einer der Aufzählungswerte [ENUM\\_CHART\\_PROPERTY\\_STRING](#) sein.

*string\_var*

[out] Variable des Typs string, die den Wert der angeforderten Eigenschaft annimmt.

### Rückgabewert

Wert des Typs string.

für die zweite Variante des Ausrufes gibt true zurück, wenn diese Eigenschaft zugaenglich wird und der Wert in der Variable string\_var aufbewahren wird, anderenfalls gibt false zurück. Für die Erhaltung der weiteren Information über den [Fehler](#), muss die Funktion [GetLastError\(\)](#) aufgerufen werden.

### Bemerkung

Funktion ChartGetString kann für Einlesen der Kommentare verwendet werden, die durch die Funktionen [Comment](#) oder [ChartSetString](#) ausgegeben werden.

### Beispiel:

```
void OnStart()
{
    ChartSetString(0, CHART_COMMENT, "Test comment.\nSecond line.\nThird!");
    ChartRedraw();
}
```



```
Sleep(1000);  
string comm=ChartGetString(0,CHART_COMMENT);  
Print(comm);  
; }
```

Sehen Sie auch

[Comment](#), [ChartSetString](#)

## ChartNavigate

Verschiebt den Chart um die angegebene Anzahl der Balken in Bezug auf die angegebene Chartposition.

```
bool ChartNavigate(
    long  chart_id,    // Identifikator des Charts
    int   position,    // Position
    int   shift=0      // Verschiebungswert
);
```

### Parameter

*chart\_id*

[in] Identifikator des Charts. 0 bedeutet den laufenden Chart.

*position*

[in] Chartposition, in Bezug auf die die Verschiebung durchgeführt wird. Wert kann einer der Aufzählungswerte [ENUM\\_CHART\\_POSITION](#) sein.

*shift=0*

[in] Anzahl der Bars, um die der Chart verschoben werden muss. Positive Verschiebung ist Verschiebung nach rechts (zum Chartende), negative Verschiebung ist Verschiebung nach links (zum Chartanfang). Null-Verschiebung ist dann sinnvoll, wenn Navigierung zum Anfang oder zum Ende des Charts durchgeführt wird.

### Rückgabewert

Gibt true im Erfolgsfall zurück, anderenfalls false.

### Beispiel:

```
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    //--- handle des aktuellen Charts erhalten
    long handle=ChartID();
    string comm="";
    if(handle>0) // wenn dies gelingt, zusätzlich einrichten
    {
        //--- Autoscroll deaktivieren
        ChartSetInteger(handle,CHART_AUTOSCROLL,false);
        //--- eine Verschiebung von der rechten Chart-Grenze setzen
        ChartSetInteger(handle,CHART_SHIFT,true);
        //--- Candlesticks ziehen
        ChartSetInteger(handle,CHART_MODE,CHART_CANDLES);
        //--- Anzeige-Modus für Tick-Volumen einstellen
        ChartSetInteger(handle,CHART_SHOW_VOLUMES,CHART_VOLUME_TICK);

        //--- einen Text vorzubereiten, um in Comment() anzuzeigen
```

```

comm="10 Balken nach rechts von Anfang der Geschichte scrollen";
//--- Kommentar anzeigen
Comment(comm);
//--- 10 Balken nach rechts von Anfang der Geschichte scrollen
ChartNavigate(handle,CHART_BEGIN,10);
//--- erhalten wir die Nummer des ersten Balken, der auf dem Chart sichtbar ist
long first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
//--- Zeilenvorschubzeichen hinzufügen
comm=comm+"\r\n";
//--- Kommentar hinzufügen
comm=comm+"Der erste Balken auf dem Chart hat Nummer "+IntegerToString(first_bar);
//--- Kommentar anzeigen
Comment(comm);
//--- 5 Sekunden warten, um zu sehen, wie das Chart bewegt
Sleep(5000);

//--- Kommentar hinzufügen
comm=comm+"\r\n"+"10 Balken nach links von der rechten Chart-Grenze scrollen";
Comment(comm);
//--- 10 Balken nach links von der rechten Chart-Grenze scrollen
ChartNavigate(handle,CHART_END,-10);
//--- erhalten wir die Nummer des ersten Balken, der auf dem Chart sichtbar ist
first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
comm=comm+"\r\n";
comm=comm+"Der erste Balken auf dem Chart hat Nummer "+IntegerToString(first_bar);
Comment(comm);
//--- 5 Sekunden warten, um zu sehen, wie das Chart bewegt
Sleep(5000);

//--- neuen Block von Chart-Scrollen
comm=comm+"\r\n"+"300 Balken nach rechts von Anfang der Geschichte scrollen";
Comment(comm);
//--- 300 Balken nach rechts von Anfang der Geschichte scrollen
ChartNavigate(handle,CHART_BEGIN,300);
first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
comm=comm+"\r\n";
comm=comm+"Der erste Balken auf dem Chart hat Nummer "+IntegerToString(first_bar);
Comment(comm);
//--- 5 Sekunden warten, um zu sehen, wie das Chart bewegt
Sleep(5000);

//--- neuen Block von Chart-Scrollen
comm=comm+"\r\n"+"300 Balken nach links von der rechten Chart-Grenze scrollen";
Comment(comm);
//--- 300 Balken nach links von der rechten Chart-Grenze scrollen
ChartNavigate(handle,CHART_END,-300);
first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
comm=comm+"\r\n";
comm=comm+"Der erste Balken auf dem Chart hat Nummer "+IntegerToString(first_bar);

```

```
    Comment(comm);  
; }  
; }
```

## ChartID

Gibt Identifikator des laufenden Charts zurück.

```
long ChartID();
```

### Rückgabewert

Wert des Typs [long](#).

## ChartIndicatorAdd

Setzt Indikator mit dem angegebenen handle in das angegebene Fenster des Charts zu. Indikator und Chart sollten auf dem gleichen Symbol und Zeitrahmen gebaut werden.

```
bool ChartIndicatorAdd(  
    long chart_id,           // Identifikator des Charts  
    int sub_window           // Nummer des Unterfensters  
    int indicator_handle     // handle des Indikators  
);
```

### Parameter

*chart\_id*

[in] Identifikator des Charts. 0 bedeutet das laufende Chart.

*sub\_window*

[in] Nummer des Unterfensters des Charts. 0 bedeutet das Hauptfenster des Charts. Um ein Indikator in ein neues Fenster hinzuzufügen, der Parameter muss um eins größer als der Index des letzten existierten Fensters sein, d.h. dem [CHART\\_WINDOWS\\_TOTAL](#) gleich sein. Wenn der Parameterwert mehr als [CHART\\_WINDOWS\\_TOTAL](#) ist, das neue Fenster wird nicht erstellt werden und der Indikator wird nicht hinzugefügt werden.

*indicator\_handle*

[in] Handle des Indikators.

### Rückgabewert

Gibt true im Erfolgsfall zurück, anderenfalls false. Um die Information über den [Fehler](#) zu erhalten, muss man die Funktion [GetLastError\(\)](#) aufrufen. Fehler 4114 bedeutet, dass der Chart und das hinzugefügte Indikator haben verschiedenen Symbol oder Zeitrahmen.

### Hinweis

Wenn ein Indikator, der in einem separaten Fenster gezeichnet werden soll (z. B. integrierte [IMACD](#) oder benutzerdefinierter Indikator mit dem angegebenen [#property indicator\\_separate\\_window](#)) in den Hauptfenster des Charts hinzugefügt wird, kann dieser Indikator unsichtbar sein, obwohl er in der Liste der Indikatoren angezeigt wird. Dies bedeutet, dass Skala des Indikators von der Skala des Charts sich unterscheidet, und Werte des Indikators sind unsichtbar in angezeigte Skala des Charts. In diesen Fall gibt [GetLastError\(\)](#) Code von Null zurück, das bedeute keinen Fehler. Die Werte dieses "unsichtbaren" Indikators wird in der "Datenfenster" sichtbar und können auch von anderen MQL5-Programme erhalten werden.

### Beispiel:

```

#property description "Expert Advisor demonstriert Funktion ChartIndicatorAdd()."
#property description "Nach dem Start auf dem Chart (und Bekommen von Fehler im Log),
#property description "Eigenschaften von Expert Advisor und geben Sie die richtigen Pa
#property description "Indikator MACD wird auf dem Chart hinzugefügt."

//--- input parameters
input string      symbol="AUDUSD";      // Symbolname
input ENUM_TIMEFRAMES period=PERIOD_M12; // Zeitrahmen
input int         fast_ema_period=12;    // Periode der raschen MACD
input int         slow_ema_period=26;    // Periode der langsamen MACD
input int         signal_period=9;      // Mittelungsperiode der Unterschied
input ENUM_APPLIED_PRICE apr=PRICE_CLOSE; // Preisetyp für Berechnen von MACD

int indicator_handle=INVALID_HANDLE;

//+-----+
//| Expert initialization function |
//+-----+
int OnInit()
{
//---
    indicator_handle=iMACD(symbol,period,fast_ema_period,slow_ema_period,signal_period,
//--- versuchen wir den Indikator zum Chart hinzuzufügen
    if(!AddIndicator())
    {
        //--- Funktion AddIndicator() weigerte sich, den Indikator zum Chart hinzuzufügen
        int answer=MessageBox("Trotzdem versuchen MACD auf dem Chart hinzufügen?",
                                "Falscher Symbol und/oder der Zeitrahmen für die Zugabe de
                                MB_YESNO // Auswahl Schaltfläche "Yes" und "No" gezeigt we
                                );
        //--- wenn Benutzer beharrt auf dem Missbrauch von ChartIndicatorAdd()
        if(answer==IDYES)
        {
            //--- erst anzeigen es im Log
            PrintFormat("Achtung! %s: Wir versuchen indikator MACD(%s/%s) auf dem Chart %
                        __FUNCTION__,symbol,EnumToString(period),_Symbol,EnumToString(_Pe
            //--- erhalten den Nummer von neuen Sub-Fenster, in dem wir versuchen, einen
            int subwindow=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
            //--- jetzt einen fehleranfälligen Versuch machen
            if(!ChartIndicatorAdd(0,subwindow,indicator_handle))
                PrintFormat("MACD konnte nicht auf dem Fenster %d des Charts hinzugefügt v
                            subwindow,GetLastError());
        }
    }
//---
    return(INIT_SUCCEEDED);
}

//+-----+
//| Expert tick function |
//+-----+
void OnTick()
{
// Expert Advisor mach nichts
}

//+-----+
//| Funktion prüft und fügt den Indikator auf den Chart zu |
//+-----+
bool AddIndicator()
{
//--- Nachricht
    string message;
//--- Prüfen wir den Symbol des Indikators und Symbol des Charts

```

```

    if(symbol!=_Symbol)
    {
        message="Demonstrieren die Verwendung der Funktion Demo_ChartIndicatorAdd():";
        message=message+"\r\n";
        message=message+"Sie können nicht den Indikator, der an anderem Symbol berechnet
        message=message+"\r\n";
        message=message+"Geben Sie den Chartsymbol in Eigenschaften des Experts ein - ";
        Alert(message);
        //--- Frühausgang, hinzufügen wir keinen Indikator zum Chart
        return false;
    }
//--- Prüfen wir Zeitrahmen des Indikators und Zeitrahmen des Charts
    if(period!=_Period)
    {
        message="Sie können nicht den Indikator, der an anderer Zeitrahmen berechnet wa
        message=message+"\r\n";
        message=message+"Geben Sie Zeitrahmen des Charts in Eigenschaften von Expert Adv
        Alert(message);
        //--- Frühausgang, hinzufügen wir keinen Indikator zum Chart
        return false;
    }
//--- alle Tests bestanden haben, Symbol und Periode des Indikators mit Chart gleich s
    if(indicator_handle==INVALID_HANDLE)
    {
        Print(__FUNCTION__," Erstellen wir MACD");
        indicator_handle=iMACD(symbol,period,fast_ema_period,slow_ema_period,signal_peri
        if(indicator_handle==INVALID_HANDLE)
        {
            Print("MACD konnte nicht erstellt werden. Fehlercode",GetLastError());
        }
    }
//--- reset Fehlercode
    ResetLastError();
//--- Fügen den Indikator auf dem Chart hinzu
    Print(__FUNCTION__," Fügen den Indikator MACD auf dem Chart hinzu");
    Print("MACD ist gebaut auf ",symbol,"/",EnumToString(period));
//--- Bekommen wir den Nummer des neuen Fenster, auf dem MACD hinzugefügt wird
    int subwindow=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
    PrintFormat("Fügen den Indikator auf dem Fenster %d des Charts hinzu",subwindow);
    if(!ChartIndicatorAdd(0,subwindow,indicator_handle))
    {
        PrintFormat("MACD konnte nicht auf dem Fenster %d des Chart hinzugefügt werden.
        subwindow,GetLastError());
    }
//--- Indikator war erfolgreich auf dem Chart hinzugefügt
    return(true);
}

```

Sehen Sie auch

[ChartIndicatorDelete\(\)](#), [ChartIndicatorName\(\)](#), [ChartIndicatorsTotal\(\)](#), [iCustom\(\)](#), [IndicatorCreate\(\)](#)



## ChartIndicatorDelete

Löscht Indikator mit angegebenem Namen aus dem angegebenen Chartfenster.

```
bool ChartIndicatorDelete(  
    long      chart_id,           // Identifikator des Charts  
    int       sub_window         // Nummer des Unterfensters  
    const string indicator_shortcode // Kurzname des Indikators  
);
```

### Parameter

*chart\_id*

[in] Identifikator des Charts. 0 bedeutet das laufende Chart.

*sub\_window*

[in] Nummer des Unterfensters des Charts. 0 bedeutet das Hauptfenster des Charts.

*const indicator\_shortcode*

[in] Kurzname des Indikators, der in [INDICATOR\\_SHORTNAME](#) Eigenschaft bei der Funktion [IndicatorSetString\(\)](#) angegeben wird. Sie könne den Kurzname bei der Funktion [ChartIndicatorName\(\)](#) ermitteln.

### Rückgabewert

Gibt true im Erfolgsfall zurück, anderenfalls false. Um die Information über den [Fehler](#) zu erhalten, muss man die Funktion [GetLastError\(\)](#) aufrufen.

### Bemerkung

Wenn es mehrere Indikatoren mit der gleichen Kurznamen in das angegebene Unterfenster des Charts gibt, wird das erste entfernt werden.

Wenn an den Werte des gelöschten Indikator andere Indikatoren auf dem gleichen Chart konstruiert sind, werden sie auch gelöscht werden.

Mann sollte nicht Indicators Kurzname mit dem Dateinamen der beim Erstellen des Indikators mit den Funktionen [iCustom\(\)](#) und [IndicatorCreate\(\)](#) gezeigt wird verwechseln. Wenn der Kurzname des Indikators nicht explizit festgelegt wird, dann bei der Kompilation wird den Namen der Quelldatei des Indikators angegeben.

Entfernung eines Indikators aus dem Chart bedeutet nicht, dass der Berechnungsteil des Indikators auch aus dem Terminalspeicher gelöscht wird. Um Handle des Indikators freizumachen, benutzen Sie die Funktion [IndicatorRelease\(\)](#).

Sie sollen richtig einen kurzen Namen des Indikators formen, der in der Eigenschaft [INDICATOR\\_SHORTNAME](#) mit Hilfe der Funktion [IndicatorSetString\(\)](#) geschrieben wird. Wir empfehlen, dass der Kurzname die Werte der Eingabeparameter des Indikators enthält, da der Indikator, der aus dem Chart gelöscht wird, in der Funktion [ChartIndicatorDelete\(\)](#) beim Kurzname identifiziert wird.

**Beispiel für das Löschen eines Indikators, wenn Initialisierung falsch war**

```

//+-----+
//|                                     Demo_ChartIndicatorDelete.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots  1
//--- plot Histogram
#property indicator_label1  "Histogram"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- input parameters
input int      first_param=1;
input int      second_param=2;
input int      third_param=3;
input bool     wrong_init=true;
//--- indicator buffers
double         HistogramBuffer[];
string         shortname;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    int res=INIT_SUCCEEDED;
    //--- Den Array HistogramBuffer zum Indikatorpuffer binden
    SetIndexBuffer(0,HistogramBuffer,INDICATOR_DATA);
    //--- Einen kurzen Namen des Indikators auf Eingabeparameter konstruieren
    shortname=StringFormat("Demo_ChartIndicatorDelete(%d,%d,%d)",
                           first_param,second_param,third_param);
    IndicatorSetString(INDICATOR_SHORTNAME,shortname);
    //--- Wenn Zwangsendung des Indikators eingegeben ist, dann Wert ungleich Null zurück
    if(wrong_init) res=INIT_FAILED;
    return(res);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
    //--- Ausgangsposition für Arbeit im Zyklus
    int start=prev_calculated-1;
    if(start<0) start=0;
    //--- Füllen Indikator-Puffer mit Werte
    for(int i=start;i<rates_total;i++)
    {

```

```

        HistogramBuffer[i]=close[i];
    }
    //--- return value of prev_calculated for next call
    return(rates_total);
}
//+-----+
//| Handler von Ereignis Deinit |
//+-----+
void OnDeinit(const int reason)
{
    PrintFormat("%s: Code vom Deinitialisierungsgrund=%d", __FUNCTION__, reason);
    if(reason==REASON_INITFAILED)
    {
        PrintFormat("Der Indikator mit dem Kurznamen %s (Datei %s) löscht sich aus dem C
        int window=ChartWindowFind();
        bool res=ChartIndicatorDelete(0,window,shortname);
        //--- Analysieren wir das Ergebnis des Aufrufs von ChartIndicatorDelete()
        if(!res)
        {
            PrintFormat("Indikator %s konnte nicht aus dem Fenster #%d gelöscht werden. Fe
                shortname,window,GetLastError());
        }
    }
}

```

#### Sehen Sie auch

[ChartIndicatorAdd\(\)](#), [ChartIndicatorName\(\)](#), [ChartIndicatorsTotal\(\)](#), [iCustom\(\)](#), [IndicatorCreate\(\)](#),  
[IndicatorSetString\(\)](#)

## ChartIndicatorGet

Gibt das Handle des Indikators mit dem angegebenen Kurznamen im angegebenen Chart-Fenster zurück.

```
int ChartIndicatorGet(  
    long      chart_id,           // ID des Charts  
    int       sub_window         // Nummer des Unterfensters  
    const string indicator_shortcode // Kurzname des Indikators  
);
```

### Parameter

*chart\_id*

[in] ID des Charts. 0 bedeutet den aktuellen Chart.

*sub\_window*

[in] Nummer des Unterfensters. 0 bedeutet das Hauptfenster des Charts.

*const indicator\_shortcode*

[in] Kurzname des Indikators, der in der Eigenschaft [INDICATOR\\_SHORTNAME](#) durch Funktion [IndicatorSetString\(\)](#) angegeben wird. Ein Kurzname des Indikators kann durch Funktion [ChartIndicatorName\(\)](#) erhalten werden.

### Rückgabewert

Gibt Indikator-Handle beim Erfolg zurück, ansonsten [INVALID\\_HANDLE](#). Um Informationen über den [Fehler](#) zu erhalten, rufen Sie die Funktion [GetLastError\(\)](#) an.

### Bemerkung

Sie müssen richtig den Kurznamen des Indikators wählen seiner Erstellung generieren. Der Name wird durch die Funktion [IndicatorSetString\(\)](#) in der Eigenschaft [INDICATOR\\_SHORTNAME](#) geschrieben werden. Es wird empfohlen, dass ein Kurznamen die Werte der Input-Parameter des Indikators enthält, da dein Indikators in der Funktion [ChartIndicatorGet\(\)](#) nach dem Kurznamen indentifiziert wird.

Ein anderer Weg, um den Indikator zu identifizieren - eine Liste seiner Parameter für einen bestimmten Handle durch der Funktion [IndicatorParameters\(\)](#) zu erhalten und dann die erhaltenen Werte zu analysieren.

### Beispiel:

```

//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    //--- Anzahl der Fenster auf dem Chart (es gibt immer mindestens ein Hauptfenster)
    int windows=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
    //--- für alle Fenster machen
    for(int w=0;w<windows;w++)
    {
        //--- Anzahl der Indikatoren in diesem Fenster/Untenfenster
        int total=ChartIndicatorsTotal(0,w);
        //--- für alle Fenster machen
        for(int i=0;i<total;i++)
        {
            //--- erhalten wir den kurzen Namen des Indikators
            string name=ChartIndicatorName(0,w,i);
            //--- erhalten wir Indikator-Handle
            int handle=ChartIndicatorGet(0,w,name);
            //--- In Log schreiben
            PrintFormat("Window=%d, index=%d, Name=%s, handle=%d",w,i,name,handle);
            //--- sicherlich befreien wir den Indikatorhandle, wenn er nicht mehr benötigt
            IndicatorRelease(handle);
        }
    }
}

```

Sehen Sie auch

[ChartIndicatorAdd\(\)](#), [ChartIndicatorName\(\)](#), [ChartIndicatorsTotal\(\)](#), [IndicatorParameters\(\)](#)

## ChartIndicatorName

Bringt Kurzname des Indikators beim Nummer in der Indikatorenliste des Chartfensters zurück.

```
string ChartIndicatorName(  
    long   chart_id,      // Identifikator des Charts  
    int     sub_window    // Nummer des Unterfensters  
    int     index         // Index des Indikators in der Liste der Indikatoren, die zu c  
);
```

### Parameter

*chart\_id*

[in] Identifikator des Charts. 0 bedeutet das laufende Chart.

*sub\_window*

[in] Nummer des Unterfensters des Charts. 0 bedeutet das Hauptfenster des Charts.

*index*

[in] Index des Indikators in der Liste der Indikatoren. Indikatoren Nummerierung beginnt bei Null, das heißt, hat der erste Indikator in der Liste Index Null. Um die Anzahl der Indikatoren in der Liste zu erhalten, benutzen Sie Funktion [ChartIndicatorsTotal\(\)](#).

### Rückgabewert

Kurzer Name des Indikators, der in der Eigenschaft [INDICATOR\\_SHORTNAME](#) mit Hilfe der Funktion [IndicatorSetString\(\)](#) eingegeben wird. Sie können den Kurzname eines Indikators mit der Funktion [ChartIndicatorName\(\)](#) erhalten. Um die Information über den [Fehler](#) zu erhalten, muss man die Funktion [GetLastError\(\)](#) aufrufen.

### Bemerkung

Mann sollte nicht Indikators Kurzname mit dem Dateinamen der beim Erstellen des Indikators mit den Funktionen [iCustom\(\)](#) und [IndicatorCreate\(\)](#) gezeigt wird verwechseln. Wenn der Kurzname des Indikators nicht explizit festgelegt wird, dann bei der Kompilation wird den Namen der Quelldatei des Indikators angegeben.

Sie sollen richtig einen kurzen Namen des Indikators formen, der in der Eigenschaft [INDICATOR\\_SHORTNAME](#) mit Hilfe der Funktion [IndicatorSetString\(\)](#) geschrieben wird. Wir empfehlen, dass der Kurzname die Werte der Eingabeparameter des Indikators enthält, da der Indikator, der aus dem Chart gelöscht wird, in der Funktion [ChartIndicatorDelete\(\)](#) beim Kurzname identifiziert wird.

### Sehen Sie auch

[ChartIndicatorAdd\(\)](#), [ChartIndicatorDelete\(\)](#), [ChartIndicatorsTotal\(\)](#), [iCustom\(\)](#), [IndicatorCreate\(\)](#), [IndicatorSetString\(\)](#)

## ChartIndicatorsTotal

Bringt die Anzahl aller Indikatoren am angegebenen Chartfenster.

```
int ChartIndicatorsTotal(  
    long chart_id,      // Identifikator des Charts  
    int sub_window      // Nummer des Unterfensters  
);
```

### Parameter

*chart\_id*

[in] Identifikator des Charts. 0 bedeutet das laufende Chart.

*sub\_window*

[in] Nummer des Unterfensters des Charts. 0 bedeutet das Hauptfenster des Charts.

### Rückgabewert

Anzahl aller Indikatoren am angegebenen Chartfenster. Um die Information über den [Fehler](#) zu erhalten, muss man die Funktion [GetLastError\(\)](#) aufrufen.

### Bemerkung

Die Funktion ermöglicht durch alle Indikatoren des Charts zu suchen. Die Anzahl aller Fenster des Charts kann aus der Eigenschaft [CHART\\_WINDOWS\\_TOTAL](#) mit der Funktion [ChartGetInteger\(\)](#) erhalten werden.

### Sehen Sie auch

[ChartIndicatorAdd\(\)](#), [ChartIndicatorDelete\(\)](#), [ChartIndicatorsTotal\(\)](#), [iCustom\(\)](#), [IndicatorCreate\(\)](#), [IndicatorSetString\(\)](#)

## ChartWindowOnDropped

Gibt die Nummer des Subfensters zurück, wohin dieser Expert oder Script mit der Maus versetzt wurde. 0 bedeutet Hauptfenster des Charts.

```
int ChartWindowOnDropped();
```

### Rückgabewert

Wert des Typs [int](#).

### Beispiel:

```
int myWindow=ChartWindowOnDropped();  
int windowsTotal=ChartGetInteger(0,CHART_WINDOWS_TOTAL);  
Print("Script ist im Fenster ausgeführt #" +myWindow+  
      ". Insgesamt Fenster auf dem Chart "+ChartSymbol()+" : ",windowsTotal);
```

### Sehen Sie auch

[ChartPriceOnDropped](#), [ChartTimeOnDropped](#), [ChartXOnDropped](#), [ChartYOnDropped](#)



## ChartPriceOnDropped

Gibt die Preiskordinate zurück, wohin dieser Expert oder Script versetzt wurde.

```
double ChartPriceOnDropped();
```

### Rückgabewert

Wert des Typs [double](#).

### Beispiel:

```
double p=ChartPriceOnDropped();  
Print("ChartPriceOnDropped() = ",p);
```

### Sehen Sie auch

[ChartXOnDropped](#), [ChartYOnDropped](#)

## ChartTimeOnDropped

Gibt die Zeitkoordinate zurück, wohin dieser Expert oder Script mit der Maus versetzt wurde.

```
datetime ChartTimeOnDropped();
```

### Rückgabewert

Wert des Typs [datetime](#).

### Beispiel:

```
datetime t=ChartTimeOnDropped();  
Print("Script wasdropped on the "+t);
```

### Sehen Sie auch

[ChartXOnDropped](#), [ChartYOnDropped](#)

## ChartXOnDropped

Gibt X-Koordinate zurück, wohin dieser Expert oder Script versetzt wurde.

```
int ChartXOnDropped();
```

### Rückgabewert

Wert der X-Koordinate.

### Bemerkung

X-Achse ist von links nach rechts gerichtet.

### Beispiel:

```
int X=ChartXOnDropped();  
int Y=ChartYOnDropped();  
Print("(X,Y) = (" + X + ", " + Y + ")");
```

### Sehen Sie auch

[ChartWindowOnDropped](#), [ChartPriceOnDropped](#), [ChartTimeOnDropped](#)

## ChartYOnDropped

Gibt Y-Koordinate zurück, wohin dieser Expert oder Script mit der Maus versetzt wurde.

```
int ChartYOnDropped();
```

### Rückgabewert

Wert der Y-Koordinate.

### Bemerkung

Y-Achse ist von oben nach unten gerichtet.

### Sehen Sie auch

[ChartWindowOnDropped](#), [ChartPriceOnDropped](#), [ChartTimeOnDropped](#)

## ChartSetSymbolPeriod

Verändert die Werte des Symbols und der Periode des angegebenen Charts. Die Funktion arbeitet asynchron, sie gibt die Befehlsanweisung und wartet auf ihre Durchführung nicht. Der Befehl wird in die Warteschlange der Chartnachrichten gestellt und wird nach der Prozessierung aller vorherigen Befehle abgearbeitet werden.

```
bool ChartSetSymbolPeriod(  
    long      chart_id,      // Identifikator des Charts  
    string     symbol,       // Symbolname  
    ENUM_TIMEFRAMES period   // Periode  
);
```

### Parameter

*chart\_id*

[in] Identifikator des Charts. 0 bedeutet den laufenden Chart.

*symbol*

[in] Symbol des Charts. [NULL](#) bedeutet Symbol des laufenden Charts (zum dem Expert angehängt wird)

*period*

[in] Periode des Charts (Timeframe). Kann einen der Aufzählungswerte [ENUM\\_TIMEFRAMES](#) annehmen. 0 bedeutet Periode des laufenden Charts.

### Rückgabewert

Gibt true zurück, wenn der Befehl in die Warteschlange des Charts gestellt wurde, andernfalls gibt false zurück. Um die Information über den [Fehler](#) zu erhalten, muss man die Funktion [GetLastError\(\)](#) aufrufen.

### Bemerkung

Veränderung des Symbols/der Periode führt zu Neuinitialisierung des Experten, der zum entsprechenden Chart angehängt wird.

### Sehen Sie auch

[ChartSymbol](#), [ChartPeriod](#)

## ChartScreenShot

Die Funktion gewährt Screenshot des angegebenen Charts im laufenden Zustand in Format GIF, PNG oder BMP je nach angegebenen Erweiterung.

```
bool ChartScreenShot(
    long          chart_id,           // Identifikator des Charts
    string        filename,          // Dateiname
    int           width,             // Breite
    int           height,            // Höhe
    ENUM_ALIGN_MODE align_mode=ALIGN_RIGHT // Typ der Ausrichtung
);
```

### Parameter

*chart\_id*

[in] Identifikator des Charts. 0 bedeutet den laufenden Chart.

*filename*

[in] Dateiname des Screenshot. Kann nicht mehr als 63 Symbole sein. Screenshot wird m Verzeichnis \Files aufbewahren.

*width*

[in] Breite des Screenshot in Pixel

*height*

[in] Hoehe des Screenshot in Pixel

*align\_mode=ALIGN\_RIGHT*

[in] Ausgabemodus des Screenshot. Wert von Aufzählung [ENUM\\_ALIGN\\_MODE](#). ALIGN\_RIGHT bedeutet Ausrichtung rechtsbündig. (Ausgabe vom Ende). ALIGN\_LEFT bedeutet Ausrichtung linksbündig.

### Rückgabewert

Bringt true im Erfolgsfall zurück, anderenfalls false.

### Bemerkung

Wenn es notwendig ist, Screenshot von der bestimmten Position zu nehmen, muss der Chart vor allem durch die Funktion [ChartNavigate\(\)](#) positioniert werden. Wenn horizontale Größe des Screenshots kleiner als Fenster des Charts ist, wird entweder der rechte Chartteil oder der linke Chartteil ausgegeben, abhängig vom Parameterwert align\_mode.

### Beispiel:

```
#property description "Der Expert Advisor zeigt, wie eine Reihe von Screenshots des al
#property description "mit der ChartScreenShot()-Funktion zu erstellen. Für die Bequer
#property description "auf den Chart angezeigt. Die Höhe und Breite von Bildern wird c

#define WIDTH 800 // Breite des Bildes für Anruf von ChartScreenShot()
#define HEIGHT 600 // Höhe des Bildes für Anruf von ChartScreenShot()
```

```

//--- input parameters
input int    pictures=5;    // Anzahl der Bilder in Serie
int         mode=-1;        // -1 bedeutet Einrückung an den rechten Rand des Chart,
int         bars_shift=300; // Anzahl der Balken beim Scroll des Charts mit ChartNav
//+-----+
//| Expert initialization function                                |
//+-----+
void OnInit()
{
//--- Autoscroll des Charts deaktivieren
    ChartSetInteger(0,CHART_AUTOSCROLL,false);
//--- Einrückung des rechten Rands des Charts angeben
    ChartSetInteger(0,CHART_SHIFT,true);
//--- Darstellung des Charts in Form von Kerzendiagramm
    ChartSetInteger(0,CHART_MODE,CHART_CANDLES);
//---
    Print("Vorbereitung des Expert Advisors für die Arbeit ist abgeschlossen");
; }
//+-----+
//| Expert tick function                                        |
//+-----+
void OnTick()
{
//---

; }
//+-----+
//| ChartEvent function                                        |
//+-----+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
{
//--- Ausgabe der Name der Funktion, der Zeit des Aufrufs und Ereignis-ID
    Print(__FUNCTION__,TimeCurrent()," id=",id," mode=",mode);
//--- Verarbeitung von Ereignissen CHARTEVENT_CLICK ("Klick mit der Maus auf dem Chart)
    if(id==CHARTEVENT_CLICK)
    {
        //--- Erste Einrückung vom Rand des Charts
        int pos=0;
        //--- Arbeit mit dem linken Rand des Charts
        if(mode>0)
        {
            //--- Den Chart nach den linken Rand scrollen
            ChartNavigate(0,CHART_BEGIN,pos);
            for(int i=0;i<pictures;i++)
            {
                //--- Die Beschriftung auf dem Chart und den Dateinamen vorbereiten

```

```

        string name="ChartScreenShot"+"CHART_BEGIN"+string(pos)+".gif";
        ///--- Ausgabe des Namen auf dem Chart als Kommentar
        Comment(name);
        ///--- Den Screenshot in der Datei in Terminal_Verzeichnis\MQL5\Files\ spe
        if(ChartScreenShot(0,name,WIDTH,HEIGHT,ALIGN_LEFT))
            Print("Gespeichertes Screenshot ",name);
        ///---
        pos+=bars_shift;
        ///--- Gib dem Benutzer Zeit, um auf neuen Abschnitt des Chart zu sehen
        Sleep(3000);
        ///--- Den Chart von der aktuellen Position um bars_shift nach rechts scrol
        ChartNavigate(0,CHART_CURRENT_POS,bars_shift);
    };
    ///--- Regimewechsel auf die gegenüberliegende
    mode*=-1;
};
else // Arbeit mit dem rechten Rand des Charts
{
    ///--- Den Chart nach den rechten Rand scrollen
    ChartNavigate(0,CHART_END,pos);
    for(int i=0;i<pictures;i++)
    {
        ///--- Die Beschriftung auf dem Chart und den Dateinamen vorbereiten
        string name="ChartScreenShot"+"CHART_END"+string(pos)+".gif";
        ///--- Ausgabe des Namen auf dem Chart als Kommentar
        Comment(name);
        ///--- Den Screenshot in der Datei in Terminal_Verzeichnis\MQL5\Files\ spe
        if(ChartScreenShot(0,name,WIDTH,HEIGHT,ALIGN_RIGHT))
            Print("Gespeichertes Screenshot ",name);
        ///---
        pos+=bars_shift;
        ///--- Gib dem Benutzer Zeit, um auf neuen Abschnitt des Chart zu sehen
        Sleep(3000);
        ///--- Den Chart von der aktuellen Position um bars_shift nach rechts scrol
        ChartNavigate(0,CHART_CURRENT_POS,-bars_shift);
    };
    ///--- Regimewechsel auf die gegenüberliegende
    mode*=-1;
};
} // Ende der Erarbeitung von CHARTEVENT_CLICK
///--- Ende von Handler OnChartEvent()
};

```

Sehen Sie auch

[ChartNavigate\(\)](#), [Ressourcen](#)



## Handelsfunktionen

Funktionsgruppe für die Verwaltung der Handelstätigkeit.

Handelsfunktionen können in Experten und Skripts benutzt werden. Handelsfunktionen können erst dann aufgerufen werden, wenn es in Eigenschaften entsprechender Experten oder Scripts das Kontrollkästchen "Ratgeber zu handeln erlauben" gibt.

Handelserlaubnis oder Handelsverbot kann von vielen Faktoren abhängig sein. Diese Faktoren sind im Abschnitt "[Handelserlaubnis](#)" beschrieben sind.

| Funktion                           | Handlung                                                                                                      |
|------------------------------------|---------------------------------------------------------------------------------------------------------------|
| <a href="#">OrderCalcMargin</a>    | Berechnet die Größe der Marge, die für den angegebenen Typ der Order notwendig ist, in der Währung des Kontos |
| <a href="#">OrderCalcProfit</a>    | Berechnet die Größe des Gewinns aufgrund der übertragenen Parameter, in der Währung des Kontos                |
| <a href="#">OrderCheck</a>         | Prüft, ob es genug Geld ist, um die notwendige <a href="#">Handelsoperation</a> auszuführen.                  |
| <a href="#">OrderSend</a>          | Sendet <a href="#">Handelsanforderungen</a> auf das Sever                                                     |
| <a href="#">OrderSendAsync</a>     | Asynchron sendet <a href="#">Handelsanforderungen</a> , ohne auf die Antwort des Handelsservers zu warten     |
| <a href="#">PositionsTotal</a>     | Bringt Anzahl der offenen Positionen zurück                                                                   |
| <a href="#">PositionGetSymbol</a>  | Bringt das Symbol der entsprechenden offenen Position zurück                                                  |
| <a href="#">PositionSelect</a>     | Wählt die offene Position für die weitere Arbeit damit                                                        |
| <a href="#">PositionGetDouble</a>  | Bringt die angeforderte Eigenschaft der offenen Position zurück (double)                                      |
| <a href="#">PositionGetInteger</a> | Bringt die angeforderte Eigenschaft der offenen Position zurück (datetime oder int)                           |
| <a href="#">PositionGetString</a>  | Bringt die angeforderte Eigenschaft der offenen Position zurück (string)                                      |
| <a href="#">OrdersTotal</a>        | Bringt die Zahl der Ordnern zurück                                                                            |
| <a href="#">OrderGetTicket</a>     | Bringt Ticket der entsprechenden Order zurück                                                                 |
| <a href="#">OrderSelect</a>        | Wählt Order für die weitere Arbeit damit                                                                      |
| <a href="#">OrderGetDouble</a>     | Bringt die angeforderte Eigenschaft der Order zurück (double)                                                 |
| <a href="#">OrderGetInteger</a>    | Bringt die angeforderte Eigenschaft der Order zurück (datetime oder int)                                      |

|                                                |                                                                                                                         |
|------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------|
| <a href="#"><u>OrderGetString</u></a>          | Bringt die angeforderte Eigenschaft der Order zurück (string)                                                           |
| <a href="#"><u>HistorySelect</u></a>           | Fordert die Deals- und Ordergeschichte für die angegebene Periode der Serverzeit an                                     |
| <a href="#"><u>HistorySelectByPosition</u></a> | Fordert die Geschichte der Deals und der Order mit dem angegebenen <a href="#"><u>Identifikator der Position</u></a> an |
| <a href="#"><u>HistoryOrderSelect</u></a>      | Wählt Order in der Geschichte für die weitere Arbeit damit                                                              |
| <a href="#"><u>HistoryOrdersTotal</u></a>      | Bringt die Orderzahl in der Geschichte zurück                                                                           |
| <a href="#"><u>HistoryOrderGetTicket</u></a>   | Bringt Ticket der entsprechenden Order in der Geschichte zurück                                                         |
| <a href="#"><u>HistoryOrderGetDouble</u></a>   | Bringt die angeforderte Eigenschaft der Order in der Geschichte zurück (double)                                         |
| <a href="#"><u>HistoryOrderGetInteger</u></a>  | Bringt die angeforderte Eigenschaft der Order in der Geschichte zurück (datetime oder int)                              |
| <a href="#"><u>HistoryOrderGetString</u></a>   | Bringt die angeforderte Eigenschaft der Order in der Geschichte zurück (string)                                         |
| <a href="#"><u>HistoryDealSelect</u></a>       | Wählt das Deal in der Geschichte für ihre weitere Anwendung durch entsprechende Funktionen                              |
| <a href="#"><u>HistoryDealsTotal</u></a>       | Bringt die Dealszahl in der Geschichte zurück                                                                           |
| <a href="#"><u>HistoryDealGetTicket</u></a>    | Wählt das Deal für die weitere Verarbeitung und bringt Ticket des Deals in der Geschichte zurück                        |
| <a href="#"><u>HistoryDealGetDouble</u></a>    | Bringt die angeforderte Eigenschaft des Deals in der Geschichte zurück (double)                                         |
| <a href="#"><u>HistoryDealGetInteger</u></a>   | Bringt die angeforderte Eigenschaft des Deals in der Geschichte zurück (datetime oder int)                              |
| <a href="#"><u>HistoryDealGetString</u></a>    | Bringt die angeforderte Eigenschaft des Deals in der Geschichte zurück (string)                                         |

## OrderCalcMargin

Berechnet die Grösse der Marge für den angegebenen Typ der Order auf dem laufenden Konto und bei der laufenden Marktumgebung, ohne Berechnung der Warteorder und der offenen Positionen. Ermöglicht die Grösse der Marge für die geplante Handelsoperation einzuschätzen. Wert wird in der Währung des Kontos zurückgegeben.

```
bool OrderCalcMargin (
    ENUM_ORDER_TYPE   action,           // Typ der Order
    string             symbol,           // Symbolname
    double             volume,           // Volumen
    double             price,            // Eröffnungspreis
    double&            margin            // Variable für Erhaltung des Wertes der Marge
);
```

### Parameter

*action*

[in] Typ der Order, kann Werte aus der Aufzählung [ENUM\\_ORDER\\_TYPE](#) annehmen.

*symbol*

[in] Name des Finanzinstrumentes.

*volume*

[in] Volumen der Handelsoperation.

*price*

[in] Eröffnungspreis.

*margin*

[out] Variable, in die die notwendige Grösse der Marge geschrieben wird, wenn die Funktion erfolgreich durchgeführt wird. Berechnung erfolgt als wäre es auf dem laufenden Konto keine Warteordern und offene Positionen. Wert der Marge hängt von vielen Faktoren ab und kann sich bei der Veränderung der Marktumgebung verändern.

### Rückgabewert

Gibt `true` im Erfolgsfall zurück, anderenfalls `false`. Für die Erhaltung der Information über den [Fehler](#), muss die Funktion [GetLastError\(\)](#) aufgerufen werden.

### Sehen Sie auch

[OrderSend\(\)](#), [Ordereigenschaften](#), [Typen der Handelsoperationen](#)

## OrderCalcProfit

Berechnet die Grösse des Gewinns für das laufende Konto und Marktumgebung aufgrund der übertragenen Parameter. Bestimmt für vorläufige Einschätzung der Handelsoperation. Der Wert wird in der Währung des Kontos zurückgegeben.

```
bool OrderCalcProfit (
    ENUM_ORDER_TYPE      action,           // Ordertyp (ORDER_TYPE_BUY oder ORDER_TYPE_SELL)
    string                symbol,           // Symbolname
    double                volume,           // Volumen
    double                price_open,        // Eröffnungspreis
    double                price_close,       // Abschlusspreis
    double&               profit             // Variable für Erhaltung des Wertes des Gewinns
);
```

### Parameter

*action*

[in] Ordertyp, kann einen der zwei Werte der Aufzählung [ENUM\\_ORDER\\_TYPE](#) annehmen: ORDER\_TYPE\_BUY oder ORDER\_TYPE\_SELL.

*symbol*

[in] Name des Finanzinstrumentes .

*volume*

[in] Volumen der Handelsoperation.

*price\_open*

[in] Eröffnungspreis.

*price\_close*

[in] Abschlusspreis.

*profit*

[out] Variable in die der berechnete Wert des Gewinns bei der erfolgreichen Ausführung der Funktion geschrieben wird. Wert der Gewinneinschätzung hängt von vielen Faktoren ab und kann sich bei der Veränderung der Marktumgebung verändern.

### Rückgabewert

Gibt true im Erfolgsfall zurück, anderenfalls false. Wenn es einen unzulässigen Ordertyp angegeben wird, gibt die Funktion false zurück. Für die Erhaltung der Information über den [Fehler](#), muss die Funktion [GetLastError\(\)](#) aufgerufen werden.

### Sehen Sie auch

[OrderSend\(\)](#), [Ordereigenschaften](#), [Typen der Handelsoperationen](#)

## OrderCheck

Funktion `OrderCheck()` prüft, ob es genug Geld ist, um die notwendige [Handelsoperationen](#) auszuführen. Ergebnisse der Prüfung werden in Felder der Struktur [MqlTradeCheckResult](#) gesetzt.

```
bool OrderCheck(  
    MqlTradeRequest&    request,    // Struktur der Anforderung  
    MqlTradeCheckResult& result    // Struktur der Antwort  
);
```

### Parameter

*request*

[in] Anzeiger auf die Struktur des Typs [MqlTradeRequest](#), die die erforderliche Handelsoperation beschreibt.

*result*

[in,out] Anzeiger auf die Struktur des Typs [MqlTradeCheckResult](#), in die das Ergebnis der Prüfung gesetzt wird.

### Rückgabewert

Wenn es an Geldmittel fehlt, oder Parameter nicht korrekt ausgefüllt sind, kehrt die Funktion `false` zurück. Bei der erfolgreichen Grundprüfung der Strukturen (Prüfung der Anzeiger) kehrt `true` zurück - das zeugt nicht davon, dass die angeforderte Handelsoperation unbedingt erfolgreich ausgeführt wird. Für die Erhaltung der ausführlichen Beschreibung der Ausführung der Funktion muss man die Felder der Struktur *result* analysieren.

Für die Erhaltung der Information über den [Fehler](#), muss man die Funktion [GetLastError\(\)](#) aufrufen.

### Sehen Sie auch

[OrderSend\(\)](#), [Typen der Handelsoperationen](#), [Struktur der Handelsanforderung](#), [Struktur der Ergebnisse der Prüfung der Handelsanforderung](#), [Struktur des Ergebnisses der Handelsanforderung](#).

## OrderSend

Funktion OrderSend() ist für die Handelsaktivitaet innerhalb MQL5 bestimmt.

```
bool OrderSend(
    MqlTradeRequest& request // Anforderungsstruktur
    MqlTradeResult& result // Antwortstruktur
);
```

### Parameter

*request*

[in] Anzeiger auf die Struktur des Typs [MqlTradeRequest](#), die Handelshandlung des Kunden beschreibt.

*result*

[in,out] Anzeiger auf die Struktur des [MqlTradeResult](#), die das Ergebnis der Handelsoperation beschreibt, wenn diese erfolgreich durchgeführt ist (kehrt zurück beim true).

### Rückgabewert

Wenn die Basenstrukturpruefung erfolgreich durchgeführt wird (Anzeigerpruefung) kommt true zurück - **das bekundet erfolgreiche Operationsausführung nicht** . Um die ausführliche Beschreibung vom Ergebnis der Funktionsdurchführung zu bekommen, muss man Felder der Struktur *result* analysieren.

### Bemerkung

Die Handelsanfrage geht durch mehrere Stadien der Prüfungen auf dem Handelsserver. Zunächst wird überprüft, ob alle benötigten Felder des Parameter *request* korrekt ausgefüllt werden. Wenn es keine Fehler gibt, akzeptiert der Server die Order für die weitere Bearbeitung. Wird die Order erfolgreich vom Handel-Server akzeptiert, gibt die Funktion OrderSend() den Wert true zurück.

Es ist empfehlenswert, die Anfrage vor dem Senden an Handel-Server selbständig zu überprüfen. Um die Anfrage zu überprüfen, verwenden Sie die Funktion [OrderCheck\(\)](#). Sie prüft, ob es genügend Mittel für die Ausführung der Handelstransaktion gibt und gibt viele nützliche Parameter in den [Ergebnissen der Prüfung von Handelsanfrage](#) zurück:

- [Rückgabekode](#), der über Fehler in der Prüfanfrage informiert;
- Wert der Bilanz nach Ausführung der Handelsoperation;
- Wert der Eigenmittel nach Ausführung der Handelsoperation;
- Wert des fließenden Gewinns nach Ausführung der Handelsoperation;
- Grösse der Marge, die für die erforderliche Handelsoperation notwendig ist;
- Grösse der freien Eigenmittel, die nach Ausführung der erforderlichen Handelsoperation bleiben;
- Level der Marge nach Ausführung der erforderlichen Handelsoperation;
- Kommentar zum Antwort-Code, Beschreibung des Fehlers.

Es ist bei der Ausstellung der Markt-Order zu beachten, dass der erfolgreiche Abschluss der Arbeit des Modus OrderSend() nicht immer eine erfolgreiche Ausführung des Deals bedeutet. Man muss in der zurückgegebenen [Struktur des Ergebnisses](#) *result* den Wert *retcode*, der den [Rückgabekode des Handel-Servers](#) enthält, und auch den Wert der Felder *deal* oder *order* je nach [Typ der Operation](#) überprüfen.

Jede angenommene Order wird auf dem Handel-Server während des Wartens auf die Bearbeitung

aufbewahrt, solange bis eine der Bedingungen für ihre Ausführung erfüllt wird:

- Gültigkeitsablauf,
- Erscheinen einer entgegenkommenden Anfrage,
- Auslösen der Order beim Erhalt von Ausführungspreis,
- Eingang einer Anfrage für die Stornierung der Order.

Zum Zeitpunkt der Orderbearbeitung sendet der Handel-Server eine Meldung an das Terminal über das Auftreten von Handelsereignis [Trade](#), das mit der Funktion [OnTrade\(\)](#) bearbeitet werden kann.

Das Ergebnis der Ausführung des Handelsanfrage, das mit der Funktion [OrderSend\(\)](#) gesendet wurde, kann mit Hilfe des Handlers [OnTradeTransaction](#) auf dem Server verfolgt werden. Dabei ist es zu beachten, dass als Ergebnis der Ausführung eines Handels-Anfrage der Handler [OnTradeTransaction](#) mehrmals aufgerufen wird.

Zum Beispiel, beim Senden Marktkauforder, wird Marktkauforder behandelt und eine entsprechende Marktkauforder für das Konto erstellt. Ist die Order ausgeführt, wird sie aus der Auftragsliste entfernt und zur Ordergeschichte hinzugefügt. Dann wird der entsprechende Deal zur Geschichte hinzugefügt und eine neue Position wird erstellt. Für jedes dieser Ereignisse wird die Funktion [OnTradeTransaction](#) aufgerufen.

#### Beispiel:

```
//--- Werte für ORDER_MAGIC
input long order_magic=55555;
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//--- vergewissern wir uns, dass der Code ein Uebungskode ist
if(AccountInfoInteger(ACCOUNT_TRADE_MODE)==ACCOUNT_TRADE_MODE_REAL)
{
Alert("Arbeit des Scripts im realen Konto ist verboten!");
return;
};
//--- stellen wir Order ein oder entfernen sie
if(GetOrdersTotalByMagic(order_magic)==0)
{
//--- keine laufende Order - stellen wir Order ein
uint res=SendRandomPendingOrder(order_magic);
Print("Rückgabekode des Handelsservers ",res);
};
else // es gibt Order - entfernen wir Order
{
DeleteAllOrdersByMagic(order_magic);
};
//---
};
//+-----+
//| Laufende Anzahl der Order mit dem angegebenen ORDER_MAGIC erhalten |
//+-----+
```

```

int GetOrdersTotalByMagic(long const magic_number)
{
    ulong order_ticket;
    int total=0;
    ///--- gehen wir durch alle wartenden Order
    for(int i=0;i<OrdersTotal();i++)
        if((order_ticket=OrderGetTicket(i))>0)
            if(magic_number==OrderGetInteger(ORDER_MAGIC)) total++;
    ///---
    return(total);
};
//+-----+
//| Entfernen wir alle wartenden Ordnern mit dem angegebenen ORDER_MAGIC |
//+-----+
void DeleteAllOrdersByMagic(long const magic_number)
{
    ulong order_ticket;
    ///---gehen wir durch alle wartenden Order
    for(int i=0;i<OrdersTotal();i++)
        if((order_ticket=OrderGetTicket(i))>0)
            ///--- Order mit dem passenden ORDER_MAGIC
            if(magic_number==OrderGetInteger(ORDER_MAGIC))
            {
                MqlTradeResult result={0};
                MqlTradeRequest request={0};
                request.order=order_ticket;
                request.action=TRADE_ACTION_REMOVE;
                OrderSend(request,result);
                ///--- geben wir in log Serverantwort aus
                Print(__FUNCTION__,":",result.comment,"Antwortkode",result.retcode);
            };
    ///---
};
//+-----+
//| Die zufaellig wartende Order einstellen |
//+-----+
uint SendRandomPendingOrder(long const magic_number)
{
    ///--- bereiten wir eine Anforderung vor
    MqlTradeRequest request={0};
    request.action=TRADE_ACTION_PENDING;           // Einstellung der wartenden Order
    request.magic=magic_number;                     // ORDER_MAGIC
    request.symbol=_Symbol;                         // Instrument
    request.volume=0.1;                             // Volumen 0.1 Lot
    request.sl=0;                                    // Stop Loss nicht angegeben
    request.tp=0;                                    // Take Profit nicht angegeben
    ///--- formieren wir Ordertyp
    request.type=GetRandomType();                   // Ordertyp
    ///---formieren wir den Preis für die wartende Order

```



```

    request.price=GetRandomPrice(request.type); // Eroffnungspreis
//--- senden wir die Handelsorder
    MqlTradeResult result={0};
    OrderSend(request,result);
//--- geben wir in log die Serverantwort aus
    Print(__FUNCTION__,":",result.comment);
    if(result.retcode==10016) Print(result.bid,result.ask,result.price);
//--- geben wir Antwortkode des Handelsservers zurück
    return result.retcode;
; }

//+-----+
//| Zufaelig Typ der wartenden Order erhalten |
//+-----+
ENUM_ORDER_TYPE GetRandomType()
{
    int t=MathRand()%4;
//--- 0<=t<4
    switch(t)
    {
        case(0):return(ORDER_TYPE_BUY_LIMIT);
        case(1):return(ORDER_TYPE_SELL_LIMIT);
        case(2):return(ORDER_TYPE_BUY_STOP);
        case(3):return(ORDER_TYPE_SELL_STOP);
    }
; }
//--- ungueltiger Wert
    return(WRONG_VALUE);
; }

//+-----+
//| Zufaelig den Preis erhalten |
//+-----+
double GetRandomPrice(ENUM_ORDER_TYPE type)
{
    int t=(int)type;
//--- Stoplevel für Symbol
    int distance=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_STOPS_LEVEL);
//--- erhalten wir Daten des letzten Ticks
    MqlTick last_tick={0};
    SymbolInfoTick(_Symbol,last_tick);
//--- berechnen wir den Preis entsprechend dem Typ
    double price;
    if(t==2 || t==5) // ORDER_TYPE_BUY_LIMIT oder ORDER_TYPE_SELL_STOP
    {
        price=last_tick.bid; // gehen wir vom Preis Bid aus
        price=price-(distance+(MathRand()%10)*5)*_Point;
    }
    else // ORDER_TYPE_SELL_LIMIT oder ORDER_TYPE_BUY_STOP
    {
        price=last_tick.ask; // gehen wir vom Preis Ask aus
        price=price+(distance+(MathRand()%10)*5)*_Point;
    }
}

```

```
    ; }  
    //---  
    return(price);  
    ; }
```

**Sehen Sie auch**

[Typen der Handelsoperationen](#), [Struktur der Handelsanforderung](#), [Struktur des Ergebnisses der Handelsanforderung](#)

## OrderSendAsync

Die OrderSendAsync()-Funktion wird für Durchführung von asynchronen [Handelsoperationen](#) ohne Warten auf eine Antwort auf eine gesendete [Anfrage](#) aus dem Server. Die Funktion wird für Hochfrequenz-Handel ausgelegt, wenn es unter den Bedingungen der Handelsalgorithmus inakzeptabel ist, Zeit für Warten auf eine Antwort aus dem Server zu verschwenden.

```
bool OrderSendAsync(  
    MqlTradeRequest& request,      // Struktur der Anfrage  
    MqlTradeResult& result        // Struktur der Antwort  
);
```

### Parameter

*request*

[in] Zeiger auf eine Struktur vom Typ [MqlTradeRequest](#), die die Handelsaktion des Kunden beschreibt.

*result*

[in,out] Zeiger auf eine Struktur vom Typ [MqlTradeResult](#), die das Ergebnis einer bei erfolgreicher Ausführung der Funktion (bei der Rückgabe true) beschreibt.

### Rückgabewert

Gibt true zurück, wenn die Anfrage an den Handel-Server gesendet ist. Wenn die Anfrage nicht gesendet wurde, wird false zurückgegeben. Bei Erfolg erhält der Antwortcode in der Variable *result* den wert [TRADE\\_RETCODE\\_PLACED](#) (Code 10008) - "eine Order wurde platziert". Die erfolgreiche Ausführung bedeutet nur die Tatsache des Versands, aber es gibt keine Garantie, dass die Anfrage auf dem Handelsserver kam und wurde zur Bearbeitung akzeptiert. Bei der Verarbeitung der empfangenen Anfrage sendet der Handelsserver eine Antwortnachricht an den Kundenterminal über die Änderung des aktuellen Status der Positionen, Ordern und Deals, die zur Erzeugung von Ereignis [Trade](#) führt.

Das Ergebnis der Ausführung des Handelsanfrage, das mit der Funktion OrderSendAsync() gesendet wurde, kann mit Hilfe des Handlers [OnTradeTransaction](#) auf dem Server verfolgt werden. Dabei ist es zu beachten, dass als Ergebnis der Ausführung eines Handels-Anfrage der Handler OnTradeTransaction mehrmals aufgerufen wird.

Zum Beispiel, beim Senden Marktkauforder, wird Marktkauforder behandelt und eine entsprechende Marktkauforder für das Konto erstellt. Ist die Order ausgeführt, wird sie aus der Auftragsliste entfernt und zur Ordergeschichte hinzugefügt. Dann wird der entsprechende Deal zur Geschichte hinzugefügt und eine neue Position wird erstellt. Für jedes dieser Ereignisse wird die Funktion OnTradeTransaction aufgerufen. Für die ausführlichen Informationen muss man die Parameter dieser Funktion analysieren:

- **trans** - in diesen Parameter wird die Struktur [MqlTradeTransaction](#) übergeben, die die auf das Handelskonto angewendete Handelstransaktion beschreibt;
- **request** - in diesen Parameter wird die Struktur [MqlTradeRequest](#), übergeben, die eine Handelsanfrage beschreibt, als Ergebnis deren Handelstransaktion ausgeführt wurde
- **result** - in diesen Parameter wird die Struktur [MqlTradeResult](#) übergeben, die das Ergebnis der Ausführung einer Handelsanfrage beschreibt.

### Bemerkung

Die Zweckbestimmung und die Parameter der Funktion sind der Funktion [OrderSend\(\)](#) ähnlich, aber anders als sie, es ist asynchron, das heißt, das sie die Programmarbeit während des Wartens auf die Funktion-Ergebnis nicht aussetzt. Vergleichen Sie die die Geschwindigkeit des Handels Operationen in dieser beiden Funktionen anhand des Beispiels in gegebenen EA.

#### Beispiel:

```
#property description "Ein Expert Advisor für Senden von Handelsanfragen"
                        " durch die Funktion OrderSendAsync().\r\n"
#property description "Verarbeitung der Handelssereignisse mit"
                        " Handler-Funktionen OnTrade() und OnTradeTransaction() ist ange
#property description "In Expert Advisor Parameters können Sie Magic Number"
                        " (eindeutiger Identifikator) "
#property description "und Nachrichtenausgabemodus in Experten Zeitschrift angeben. St

/-- input parameters
input int   MagicNumber=1234567;           // Expert Advisor ID
input bool  DescriptionModeFull=true;      // Mode der detaillierten Ausgabe
/-- eine Variable für Nutzung im Aufruf von HistorySelect()
datetime history_start;
//+-----+
//| Expert initialization function          |
//+-----+

int OnInit()
{
    /-- Erlaubnis für den automatisierten Handel überprüfen
    if(!TerminalInfoInteger(TERMINAL_TRADE_ALLOWED))
    {
        Alert("Automatisches Handle ist im Terminal verboten, EA wird gelöscht werden");
        ExpertRemove();
        return(-1);
    }

    /-- Handle auf einem Live-Konto ist nicht erlaubt
    if(AccountInfoInteger(ACCOUNT_TRADE_MODE)==ACCOUNT_TRADE_MODE_REAL)
    {
        Alert("Dem Expert Advisor ist nicht erlaubt auf einem Live-Konto zu handeln!");
        ExpertRemove();
        return(-2);
    }

    /-- ob es möglich ist, auf dem Konto zu handeln (zB ist es nicht erlaubt mit einem
    if(!AccountInfoInteger(ACCOUNT_TRADE_ALLOWED))
    {
        Alert("Der Handel auf dem Konto ist nicht erlaubt");
        ExpertRemove();
        return(-3);
    }

    /-- Speichern die Laufzeit des Experts für den Handelsgeschichte
    history_start=TimeCurrent();
    /--
    CreateBuySellButtons();
```

```

    return(INIT_SUCCEEDED);
}

//+-----+
//| Expert deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
    //--- graphische Objekte löschen
    ObjectDelete(0,"Buy");
    ObjectDelete(0,"Sell");
    //---
}

//+-----+
//| TradeTransaction function |
//+-----+
void OnTradeTransaction(const MqlTradeTransaction &trans,
                        const MqlTradeRequest &request,
                        const MqlTradeResult &result)
{
    //--- Titel nach dem Namen der Handelssereignisse-Handler-Funktion
    Print("> ",__FUNCTION__," at ",TimeToString(TimeCurrent(),TIME_SECONDS));
    //--- Typ der Transaktion in Form von Aufzählungswerte erhalten
    ENUM_TRADE_TRANSACTION_TYPE type=trans.type;
    //--- wenn die Transaktion ist ein Ergebnis der Verarbeitung einer Anfrage
    if(type==TRADE_TRANSACTION_REQUEST)
    {
        //---den Name der Transaktion ausgeben
        Print(EnumToString(type));
        //--- dann geben die String-Beschreibung der verarbeiteten Anfrage
        Print("-----RequestDescription\r\n",
              RequestDescription(request,DescriptionModeFull));
        //--- und anzeigen die Beschreibung des Abfrageergebnisses
        Print("----- ResultDescription\r\n",
              TradeResultDescription(result,DescriptionModeFull));
    }
    else // Für die Transaktionen anderen Typen geben wir eine vollständige Beschreibung
    {
        Print("----- TransactionDescription\r\n",
              TransactionDescription(trans,DescriptionModeFull));
    }
    //---
}

//+-----+
//| Trade function |
//+-----+
void OnTrade()
{
    //--- Statische Mitglieder, um den Kontostand zu speichern
    static int prev_positions=0,prev_orders=0,prev_deals=0,prev_history_orders=0;

```

```

//--- Handelsgeschichte anfragen
bool update=HistorySelect(history_start,TimeCurrent());
PrintFormat("HistorySelect(%s , %s) = %s",
            TimeToString(history_start),TimeToString(TimeCurrent()),(string)update);
//--- Titel nach dem Namen der Handelssereignisse-Handler-Funktion
Print("=> ",__FUNCTION__," at ",TimeToString(TimeCurrent(),TIME_SECONDS));
//--- Name des Handlers und die Anzahl der Order in der Zeit der Verarbeitung ausgeben
int curr_positions=PositionsTotal();
int curr_orders=OrdersTotal();
int curr_deals=HistoryOrdersTotal();
int curr_history_orders=HistoryDealsTotal();
//--- Anzahl der Aufträge, Positionen, Transaktionen und Änderungen in Klammern anzeigen
PrintFormat("PositionsTotal() = %d (%+d)",
            curr_positions,(curr_positions-prev_positions));
PrintFormat("OrdersTotal() = %d (%+d)",
            curr_orders,curr_orders-prev_orders);
PrintFormat("HistoryOrdersTotal() = %d (%+d)",
            curr_deals,curr_deals-prev_deals);
PrintFormat("HistoryDealsTotal() = %d (%+d)",
            curr_history_orders,curr_history_orders-prev_history_orders);
//--- Zeilenumbruch einfügen zum einfachen Ablesen des Journal
Print("");
//--- den Zustand des Kontos speichern
prev_positions=curr_positions;
prev_orders=curr_orders;
prev_deals=curr_deals;
prev_history_orders=curr_history_orders;
//---
}
//+-----+
//| ChartEvent function |
//+-----+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
{
//--- Verarbeitung von Ereignissen CHARTEVENT_CLICK ("Klick mit der Maus auf dem Chart
if(id==CHARTEVENT_OBJECT_CLICK)
{
    Print("=> ",__FUNCTION__,": sparam = ",sparam);
    //--- Mindestvolumen für Transaktion
    double volume_min=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MIN);
    //--- wenn die Schaltfläche "Buy" ist gedrückt, dann kaufen wir
    if(sparam=="Buy")
    {
        PrintFormat("Buy %s %G lot",_Symbol,volume_min);
        BuyAsync(volume_min);
        //--- Undrücken die gedrückte Schaltfläche
    }
}
}

```

```

        ObjectSetInteger(0,"Buy",OBJPROP_STATE,false);
    }
    //--- wenn die Schaltfläche "Sell" ist gedrückt, dann verkaufen wir
    if(sparam=="Sell")
    {
        PrintFormat("Sell %s %G lot",_Symbol,volume_min);
        SellAsync(volume_min);
        //--- Undrücken die gedrückte Schaltfläche
        ObjectSetInteger(0,"Sell",OBJPROP_STATE,false);
    }
    ChartRedraw();
}

//---
}

//+-----+
//| Gibt eine textuelle Beschreibung der Transaktion zurück |
//+-----+
string TransactionDescription(const MqlTradeTransaction &trans,
                             const bool detailed=true)
{
    //--- eine Zeichenfolge für Zurückgabe aus der Funktion vorbereiten
    string desc=EnumToString(trans.type)+"\r\n";
    //--- im detaillierten Modus fügen wir maximale Information hinzu
    if(detailed)
    {
        desc+="Symbol: "+trans.symbol+"\r\n";
        desc+="Deal ticket: "+(string)trans.deal+"\r\n";
        desc+="Deal type: "+EnumToString(trans.deal_type)+"\r\n";
        desc+="Order ticket: "+(string)trans.order+"\r\n";
        desc+="Order type: "+EnumToString(trans.order_type)+"\r\n";
        desc+="Order state: "+EnumToString(trans.order_state)+"\r\n";
        desc+="Order time type: "+EnumToString(trans.time_type)+"\r\n";
        desc+="Order expiration: "+TimeToString(trans.time_expiration)+"\r\n";
        desc+="Price: "+StringFormat("%G",trans.price)+"\r\n";
        desc+="Price trigger: "+StringFormat("%G",trans.price_trigger)+"\r\n";
        desc+="Stop Loss: "+StringFormat("%G",trans.price_sl)+"\r\n";
        desc+="Take Profit: "+StringFormat("%G",trans.price_tp)+"\r\n";
        desc+="Volume: "+StringFormat("%G",trans.volume)+"\r\n";
    }
    //--- Geben wir den resultierenden String zurück
    return desc;
}

//+-----+
//| Gibt eine textuelle Beschreibung der Handelsanfrage zurück |
//+-----+
string RequestDescription(const MqlTradeRequest &request,
                          const bool detailed=true)
{
    //--- eine Zeichenfolge für Zurückgabe aus der Funktion vorbereiten

```

```

    string desc=EnumToString(request.action)+"\r\n";
//--- im detaillierten Modus fügen wir maximale Information hinzu
    if(detailed)
    {
        desc+="Symbol: "+request.symbol+"\r\n";
        desc+="Magic Number: "+StringFormat("%d",request.magic)+"\r\n";
        desc+="Order ticket: "+(string)request.order+"\r\n";
        desc+="Order type: "+EnumToString(request.type)+"\r\n";
        desc+="Order filling: "+EnumToString(request.type_filling)+"\r\n";
        desc+="Order time type: "+EnumToString(request.type_time)+"\r\n";
        desc+="Order expiration: "+TimeToString(request.expiration)+"\r\n";
        desc+="Price: "+StringFormat("%G",request.price)+"\r\n";
        desc+="Deviation points: "+StringFormat("%G",request.deviation)+"\r\n";
        desc+="Stop Loss: "+StringFormat("%G",request.sl)+"\r\n";
        desc+="Take Profit: "+StringFormat("%G",request.tp)+"\r\n";
        desc+="Stop Limit: "+StringFormat("%G",request.stoplimit)+"\r\n";
        desc+="Volume: "+StringFormat("%G",request.volume)+"\r\n";
        desc+="Comment: "+request.comment+"\r\n";
    }
//--- Geben wir den den resultierenden String zurück
    return desc;
}
//+-----+
//| Gibt textuelle Beschreibung der Ergebnis der Anfrageverarbeitung |
//+-----+
string TradeResultDescription(const MqlTradeResult &result,
                             const bool detailed=true)
{
//--- eine Zeichenfolge für Zurückgabe aus der Funktion vorbereiten
    string desc="Retcode "+(string)result.retcode+"\r\n";
//--- im detaillierten Modus fügen wir maximale Information hinzu
    if(detailed)
    {
        desc+="Request ID: "+StringFormat("%d",result.request_id)+"\r\n";
        desc+="Order ticket: "+(string)result.order+"\r\n";
        desc+="Deal ticket: "+(string)result.deal+"\r\n";
        desc+="Volume: "+StringFormat("%G",result.volume)+"\r\n";
        desc+="Price: "+StringFormat("%G",result.price)+"\r\n";
        desc+="Ask: "+StringFormat("%G",result.ask)+"\r\n";
        desc+="Bid: "+StringFormat("%G",result.bid)+"\r\n";
        desc+="Comment: "+result.comment+"\r\n";
    }
//--- Geben wir den den resultierenden String zurück
    return desc;
}
//+-----+
//| Erstellt zwei Schaltflächen für Kauf und Verkauf |
//+-----+
void CreateBuySellButtons()

```



```

{
//--- überprüfen wir ob ein namens "Buy" existiert
if(ObjectFind(0,"Buy")>=0)
{
//--- wenn das gefundene Objekt keine Schaltfläche ist, löschen wir es
if(ObjectGetInteger(0,"Buy",OBJPROP_TYPE)!=OBJ_BUTTON)
ObjectDelete(0,"Buy");
}
else
ObjectCreate(0,"Buy",OBJ_BUTTON,0,0,0); // Schaltfläche "Buy" erstellen
//--- Schaltfläche "Buy" abstimmen
ObjectSetInteger(0,"Buy",OBJPROP_CORNER,CORNER_RIGHT_UPPER);
ObjectSetInteger(0,"Buy",OBJPROP_XDISTANCE,100);
ObjectSetInteger(0,"Buy",OBJPROP_YDISTANCE,50);
ObjectSetInteger(0,"Buy",OBJPROP_XSIZE,70);
ObjectSetInteger(0,"Buy",OBJPROP_YSIZE,30);
ObjectSetString(0,"Buy",OBJPROP_TEXT,"Buy");
ObjectSetInteger(0,"Buy",OBJPROP_COLOR,clrRed);
//--- überprüfen wir ob ein namens "Sell" existiert
if(ObjectFind(0,"Sell")>=0)
{
//--- wenn das gefundene Objekt keine Schaltfläche ist, löschen wir es
if(ObjectGetInteger(0,"Sell",OBJPROP_TYPE)!=OBJ_BUTTON)
ObjectDelete(0,"Sell");
}
else
ObjectCreate(0,"Sell",OBJ_BUTTON,0,0,0); // Schaltfläche "Sell" erstellen
//--- Schaltfläche "Sell" abstimmen
ObjectSetInteger(0,"Sell",OBJPROP_CORNER,CORNER_RIGHT_UPPER);
ObjectSetInteger(0,"Sell",OBJPROP_XDISTANCE,100);
ObjectSetInteger(0,"Sell",OBJPROP_YDISTANCE,100);
ObjectSetInteger(0,"Sell",OBJPROP_XSIZE,70);
ObjectSetInteger(0,"Sell",OBJPROP_YSIZE,30);
ObjectSetString(0,"Sell",OBJPROP_TEXT,"Sell");
ObjectSetInteger(0,"Sell",OBJPROP_COLOR,clrBlue);
//--- Zwangsaktualisierung des Charts um die Schaltflächen sofort zu zeichnen
ChartRedraw();
//---
}
//+-----+
//| Kauf über die asynchrone Funktion OrderSendAsync() |
//+-----+
void BuyAsync(double volume)
{
//--- Anfrage bereiten
MqlTradeRequest req={0};
req.action      =TRADE_ACTION_DEAL;
req.symbol      =_Symbol;
req.magic       =MagicNumber;

```

```

req.volume      =0.1;
req.type        =ORDER_TYPE_BUY;
req.price       =SymbolInfoDouble(req.symbol,SYMBOL_ASK);
req.deviation   =10;
req.comment     ="Buy using OrderSendAsync()";
MqlTradeResult  res={0};
if(!OrderSendAsync(req,res))
{
    Print(__FUNCTION__,": Fehler ",GetLastError()," , retcode = ",res.retcode);
}
//---
}
//+-----+
//| Verkauf über die asynchrone Funktion OrderSendAsync() |
//+-----+
void SellAsync(double volume)
{
    //--- Anfrage bereiten
    MqlTradeRequest req={0};
    req.action      =TRADE_ACTION_DEAL;
    req.symbol      =_Symbol;
    req.magic       =MagicNumber;
    req.volume      =0.1;
    req.type        =ORDER_TYPE_SELL;
    req.price       =SymbolInfoDouble(req.symbol,SYMBOL_BID);
    req.deviation   =10;
    req.comment     ="Sell using OrderSendAsync()";
    MqlTradeResult  res={0};
    if(!OrderSendAsync(req,res))
    {
        Print(__FUNCTION__,": Fehler ",GetLastError()," , retcode = ",res.retcode);
    }
    //---
}
//+-----+

```

Beispiel der Ausgabemeldungen in Protokoll "Experten":

```

12:52:52 ExpertAdvisor (EURUSD,H1) => OnChartEvent: sparam = Sell
12:52:52 ExpertAdvisor (EURUSD,H1) Sell EURUSD 0.01 lot
12:52:52 ExpertAdvisor (EURUSD,H1) => OnTradeTransaction at 09:52:53
12:52:52 ExpertAdvisor (EURUSD,H1) TRADE_TRANSACTION_REQUEST
12:52:52 ExpertAdvisor (EURUSD,H1) -----RequestDescription
12:52:52 ExpertAdvisor (EURUSD,H1) TRADE_ACTION_DEAL
12:52:52 ExpertAdvisor (EURUSD,H1) Symbol: EURUSD
12:52:52 ExpertAdvisor (EURUSD,H1) Magic Number: 1234567
12:52:52 ExpertAdvisor (EURUSD,H1) Order ticket: 16361998
12:52:52 ExpertAdvisor (EURUSD,H1) Order type: ORDER_TYPE_SELL
12:52:52 ExpertAdvisor (EURUSD,H1) Order filling: ORDER_FILLING_FOK
12:52:52 ExpertAdvisor (EURUSD,H1) Order time type: ORDER_TIME_GTC
12:52:52 ExpertAdvisor (EURUSD,H1) Order expiration: 1970.01.01 00:00
12:52:52 ExpertAdvisor (EURUSD,H1) Price: 1.29313
12:52:52 ExpertAdvisor (EURUSD,H1) Deviation points: 10
12:52:52 ExpertAdvisor (EURUSD,H1) Stop Loss: 0
12:52:52 ExpertAdvisor (EURUSD,H1) Take Profit: 0
12:52:52 ExpertAdvisor (EURUSD,H1) Stop Limit: 0
12:52:52 ExpertAdvisor (EURUSD,H1) Volume: 0.1
12:52:52 ExpertAdvisor (EURUSD,H1) Comment: Sell using OrderSendAsync()
12:52:52 ExpertAdvisor (EURUSD,H1) ----- ResultDescription
12:52:52 ExpertAdvisor (EURUSD,H1) Retcode 10009
12:52:52 ExpertAdvisor (EURUSD,H1) Request ID: 2
12:52:52 ExpertAdvisor (EURUSD,H1) Order ticket: 16361998
12:52:52 ExpertAdvisor (EURUSD,H1) Deal ticket: 15048668
12:52:52 ExpertAdvisor (EURUSD,H1) Volume: 0.1
12:52:52 ExpertAdvisor (EURUSD,H1) Price: 1.29313
12:52:52 ExpertAdvisor (EURUSD,H1) Ask: 1.29319
12:52:52 ExpertAdvisor (EURUSD,H1) Bid: 1.29313
12:52:52 ExpertAdvisor (EURUSD,H1) Comment:
12:52:52 ExpertAdvisor (EURUSD,H1) HistorySelect( 09:34 , 09:52) = true
12:52:52 ExpertAdvisor (EURUSD,H1) => OnTrade at 09:52:53
12:52:52 ExpertAdvisor (EURUSD,H1) PositionsTotal() = 1 (+1)
12:52:52 ExpertAdvisor (EURUSD,H1) OrdersTotal() = 0 (+0)
12:52:52 ExpertAdvisor (EURUSD,H1) HistoryOrdersTotal() = 2 (+2)
12:52:52 ExpertAdvisor (EURUSD,H1) HistoryDealsTotal() = 2 (+2)
12:52:52 ExpertAdvisor (EURUSD,H1) => OnTradeTransaction at 09:52:53
12:52:52 ExpertAdvisor (EURUSD,H1) ----- TransactionDescription
12:52:52 ExpertAdvisor (EURUSD,H1) TRADE_TRANSACTION_ORDER_ADD
12:52:52 ExpertAdvisor (EURUSD,H1) Symbol: EURUSD
12:52:52 ExpertAdvisor (EURUSD,H1) Deal ticket: 0
12:52:52 ExpertAdvisor (EURUSD,H1) Deal type: DEAL_TYPE_BUY
12:52:52 ExpertAdvisor (EURUSD,H1) Order ticket: 16361998
12:52:52 ExpertAdvisor (EURUSD,H1) Order type: ORDER_TYPE_SELL
12:52:52 ExpertAdvisor (EURUSD,H1) Order state: ORDER_STATE_STARTED
12:52:52 ExpertAdvisor (EURUSD,H1) Order time type: ORDER_TIME_GTC
12:52:52 ExpertAdvisor (EURUSD,H1) Order expiration: 1970.01.01 00:00
12:52:52 ExpertAdvisor (EURUSD,H1) Price: 1.29313
12:52:52 ExpertAdvisor (EURUSD,H1) Price trigger: 0
12:52:52 ExpertAdvisor (EURUSD,H1) Stop Loss: 0
12:52:52 ExpertAdvisor (EURUSD,H1) Take Profit: 0
12:52:52 ExpertAdvisor (EURUSD,H1) Volume: 0.1
12:52:52 ExpertAdvisor (EURUSD,H1) => OnTradeTransaction at 09:52:53
12:52:52 ExpertAdvisor (EURUSD,H1) ----- TransactionDescription
12:52:52 ExpertAdvisor (EURUSD,H1) TRADE_TRANSACTION_ORDER_DELETE
12:52:52 ExpertAdvisor (EURUSD,H1) Symbol: EURUSD
12:52:52 ExpertAdvisor (EURUSD,H1) Deal ticket: 0

```

```

12:52:52 ExpertAdvisor (EURUSD,H1) Deal type: DEAL_TYPE_BUY
12:52:52 ExpertAdvisor (EURUSD,H1) Order ticket: 16361998
12:52:52 ExpertAdvisor (EURUSD,H1) Order type: ORDER_TYPE_SELL
12:52:52 ExpertAdvisor (EURUSD,H1) Order state: ORDER_STATE_STARTED
12:52:52 ExpertAdvisor (EURUSD,H1) Order time type: ORDER_TIME_GTC
12:52:52 ExpertAdvisor (EURUSD,H1) Order expiration: 1970.01.01 00:00
12:52:52 ExpertAdvisor (EURUSD,H1) Price: 1.29313
12:52:52 ExpertAdvisor (EURUSD,H1) Price trigger: 0
12:52:52 ExpertAdvisor (EURUSD,H1) Stop Loss: 0
12:52:52 ExpertAdvisor (EURUSD,H1) Take Profit: 0
12:52:52 ExpertAdvisor (EURUSD,H1) Volume: 0.1
12:52:52 ExpertAdvisor (EURUSD,H1) HistorySelect( 09:34 , 09:52) = true
12:52:52 ExpertAdvisor (EURUSD,H1) => OnTrade at 09:52:53
12:52:52 ExpertAdvisor (EURUSD,H1) PositionsTotal() = 1 (+0)
12:52:52 ExpertAdvisor (EURUSD,H1) OrdersTotal() = 0 (+0)
12:52:52 ExpertAdvisor (EURUSD,H1) HistoryOrdersTotal() = 2 (+0)
12:52:52 ExpertAdvisor (EURUSD,H1) HistoryDealsTotal() = 2 (+0)
12:52:52 ExpertAdvisor (EURUSD,H1) => OnTradeTransaction at 09:52:53
12:52:52 ExpertAdvisor (EURUSD,H1) ----- TransactionDescription
12:52:52 ExpertAdvisor (EURUSD,H1) TRADE_TRANSACTION_HISTORY_ADD
12:52:52 ExpertAdvisor (EURUSD,H1) Symbol: EURUSD
12:52:52 ExpertAdvisor (EURUSD,H1) Deal ticket: 0
12:52:52 ExpertAdvisor (EURUSD,H1) Deal type: DEAL_TYPE_BUY
12:52:52 ExpertAdvisor (EURUSD,H1) Order ticket: 16361998
12:52:52 ExpertAdvisor (EURUSD,H1) Order type: ORDER_TYPE_SELL
12:52:52 ExpertAdvisor (EURUSD,H1) Order state: ORDER_STATE_FILLED
12:52:52 ExpertAdvisor (EURUSD,H1) Order time type: ORDER_TIME_GTC
12:52:52 ExpertAdvisor (EURUSD,H1) Order expiration: 1970.01.01 00:00
12:52:52 ExpertAdvisor (EURUSD,H1) Price: 1.29313
12:52:52 ExpertAdvisor (EURUSD,H1) Price trigger: 0
12:52:52 ExpertAdvisor (EURUSD,H1) Stop Loss: 0
12:52:52 ExpertAdvisor (EURUSD,H1) Take Profit: 0
12:52:52 ExpertAdvisor (EURUSD,H1) Volume: 0
12:52:52 ExpertAdvisor (EURUSD,H1) HistorySelect( 09:34 , 09:52) = true
12:52:52 ExpertAdvisor (EURUSD,H1) => OnTrade at 09:52:53
12:52:52 ExpertAdvisor (EURUSD,H1) PositionsTotal() = 1 (+0)
12:52:52 ExpertAdvisor (EURUSD,H1) OrdersTotal() = 0 (+0)
12:52:52 ExpertAdvisor (EURUSD,H1) HistoryOrdersTotal() = 2 (+0)
12:52:52 ExpertAdvisor (EURUSD,H1) HistoryDealsTotal() = 2 (+0)
12:52:52 ExpertAdvisor (EURUSD,H1) => OnTradeTransaction at 09:52:53
12:52:52 ExpertAdvisor (EURUSD,H1) ----- TransactionDescription
12:52:52 ExpertAdvisor (EURUSD,H1) TRADE_TRANSACTION_DEAL_ADD
12:52:52 ExpertAdvisor (EURUSD,H1) Symbol: EURUSD
12:52:52 ExpertAdvisor (EURUSD,H1) Deal ticket: 15048668
12:52:52 ExpertAdvisor (EURUSD,H1) Deal type: DEAL_TYPE_SELL
12:52:52 ExpertAdvisor (EURUSD,H1) Order ticket: 16361998
12:52:52 ExpertAdvisor (EURUSD,H1) Order type: ORDER_TYPE_BUY
12:52:52 ExpertAdvisor (EURUSD,H1) Order state: ORDER_STATE_STARTED
12:52:52 ExpertAdvisor (EURUSD,H1) Order time type: ORDER_TIME_GTC
12:52:52 ExpertAdvisor (EURUSD,H1) Order expiration: 1970.01.01 00:00
12:52:52 ExpertAdvisor (EURUSD,H1) Price: 1.29313
12:52:52 ExpertAdvisor (EURUSD,H1) Price trigger: 0
12:52:52 ExpertAdvisor (EURUSD,H1) Stop Loss: 0
12:52:52 ExpertAdvisor (EURUSD,H1) Take Profit: 0
12:52:52 ExpertAdvisor (EURUSD,H1) Volume: 0.1
12:52:52 ExpertAdvisor (EURUSD,H1) HistorySelect( 09:34 , 09:52) = true

```

```
12:52:52   ExpertAdvisor (EURUSD,H1)   => OnTrade at 09:52:53
12:52:52   ExpertAdvisor (EURUSD,H1)   PositionsTotal() = 1 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   OrdersTotal() = 0 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryOrdersTotal() = 2 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)   HistoryDealsTotal() = 2 (+0)
12:52:52   ExpertAdvisor (EURUSD,H1)
```

## PositionsTotal

Gibt die Anzahl der offenen Positionen zurück.

```
int PositionsTotal();
```

### Rückgabewert

Wert der Art [int](#).

### Bemerkung

Für jedes Symbol kann [Symbol](#) zu jedem Zeitpunkt nur eine [Position](#) geöffnet werden, die das Ergebnis eines oder mehrerer [Deals](#) ist. Man muss nicht Positionen und geltende [Warteordern](#) verwechseln, die auch in der Registerkarte "Handel" in der Werkzeugleiste "Werkzeuge" dargestellt werden.

Die Gesamtanzahl der Positionen auf dem [Handelskonto](#) kann nicht mehr als die [Gesamtanzahl der Finanzinstrumente](#) sein.

### Sehen Sie auch

[PositionGetSymbol\(\)](#), [PositionSelect\(\)](#), [Eigenschaften der Positionen](#)

## PositionGetSymbol

Gibt das Symbol der entsprechenden offenen Position zurück und wählt automatisch die Position für die weitere Arbeit damit mittels der Funktionen [PositionGetDouble](#), [PositionGetInteger](#), [PositionGetString](#) aus.

```
string PositionGetSymbol (
    int index      // Nummer in Positionenliste
);
```

### Parameter

*index*

[in] Positionsnummer in der Liste offener Positionen.

### Rückgabewert

Wert der Art [string](#). Wenn die Position nicht gefunden wird, gibt einen leeren String zurück. Um [Fehlercode](#) zu erhalten, rufen Sie die Funktion [GetLastError\(\)](#).

### Bemerkung

Für jedes [Symbol](#) kann zu jedem Zeitpunkt nur eine [Position](#) geöffnet werden, die als Ergebnis eines oder mehrerer [Deals](#) ist. Man muss nicht Positionen und [Warteordern](#) verwechseln, die auch in der Registerkarte "Handel" in der Werkzeugleiste "Instrumente" dargestellt werden.

Die Gesamtanzahl der Positionen auf dem [Handelskonto](#) kann nicht mehr als die [Gesamtanzahl der Finanzinstrumente](#) sein.

### Sehen Sie auch

[PositionsTotal\(\)](#), [PositionSelect\(\)](#), [Eigenschaften der Positionen](#)

## PositionSelect

Wähl eine offene Position für die weitere Arbeit damit. Gibt true zurück, wenn die Durchführung der Funktion erfolgreich beendet wird. Gibt false zurück wenn die Durchführung der Funktion erfolglos beendet wird. Für die Erhaltung der fehlerbezogenen Information, rufen Sie die Funktion [GetLastError\(\)](#) auf.

```
bool PositionSelect(  
    string symbol,      // Werkzeugname  
);
```

### Parameter

*symbol*

[in] Name des Finanzwerkzeuges.

### Rückgabewert

Wert der Art bool.

### Bemerkung

Für jedes [Symbol](#) kann zu jedem Zeitpunkt nur eine [Position](#) geöffnet werden, die das Ergebnis eines oder mehrerer [Deals](#) ist. Man muss nicht Positionen und [Warteordern](#), verwechseln, die auch in der Registerkarte "Handel" in der Werkzeugleiste "Instrumente" des Kundenterminals dargestellt werden.

Die Funktion PositionSelect() kopiert die Daten über Position ins Programmumfeld und weitere Aufrufe [PositionGetDouble\(\)](#), [PositionGetInteger\(\)](#) und [PositionGetString\(\)](#) geben die früher kopierten Daten zurück. Das bedeutet, dass die Position selbst nicht mehr sein kann (oder ihr Volumen, ihre Richtung usw. haben sich verändert), aber die Daten dieser Position können erhalten werden. Für bestimmte Erhaltung der frischen Daten über die Position ist es empfehlenswert, die Funktion PositionSelect() aufzurufen, bevor man frische Daten aufruft.

### Sehen Sie auch

[PositionGetSymbol\(\)](#), [PositionsTotal\(\)](#), [Eigenschaften der Positionen](#)



## PositionGetDouble

Funktion gibt die angeforderte Eigenschaft der offenen Position zurück, die vorläufig mittels der Funktion [PositionGetSymbol](#) oder [PositionSelect](#) ausgewählt wird. Positionseigenschaft muss der Art double sein. Es gibt 2 Varianten der Funktion.

1. Gibt den Wert der Eigenschaft sofort zurück.

```
double PositionGetDouble (
    ENUM_POSITION_PROPERTY_DOUBLE property_id    // Identifikator der Eigenschaft
);
```

2. Gibt true oder false zurück abhängig davon, ob die Funktion erfolgreich durchgeführt wird. Bei erfolgreicher Durchführung wird der Wert der Eigenschaft in die Empfangsvariable gesetzt, die durch Referenz vom letzten Parameter übertragen wird.

```
bool PositionGetDouble (
    ENUM_POSITION_PROPERTY_DOUBLE property_id,    // Identifikator der Eigenschaft
    double& double_var                          // hier nehmen wir den Wert der Eigenschaft
);
```

### Parameter

*property\_id*

[in] Identifikator der Positionseigenschaft. Der Wert kann einer der Aufzählungswerte [ENUM\\_POSITION\\_PROPERTY\\_DOUBLE](#) sein.

*double\_var*

[out] Variable der Art double, die den Wert der angeforderten Eigenschaft annimmt.

### Rückgabewert

Wert der Art [double](#).

### Bemerkung

Für jedes [Symbol](#) kann zu jedem Zeitpunkt nur eine [Position](#) geöffnet werden, die das Ergebnis eines oder mehrerer [Deals](#) ist. Man muss nicht Positionen und geltende [Warteordern](#) verwechseln, die auch in der Registerkarte "Handel" in der Werkzeugleiste "Instrumente" des Kundenterminals dargestellt werden.

Für bestimmte Erhaltung der frischen Daten über die Position ist es empfehlenswert, die Funktion [PositionSelect\(\)](#) aufzurufen, bevor man frische Daten aufruft.

### Sehen Sie auch

[PositionGetSymbol\(\)](#), [PositionSelect\(\)](#), [Eigenschaften der Positionen](#)

## PositionGetInteger

Funktion gibt die angeforderte Eigenschaft der offenen Position zurück, vorher gewählt durch die Funktionen [PositionGetSymbol](#) oder [PositionSelect](#). Positionseigenschaft muss der Art , int. sein. Es gibt zwei Arten der Funktion.

1. Gibt Wert der Eigenschaft direkt zurück.

```
long PositionGetInteger (
    ENUM_POSITION_PROPERTY_INTEGER property_id      // Eigenschaftsidentifikator
);
```

2. Gibt true oder false zurück abhängig davon, ob die Funktion erfolgreich durchgeführt wurde. Bei erfolgreicher Durchführung wird der Wert der Eigenschaft in die Empfangsvariable gesetzt, die durch Referenz vom letzten Parameter übertragen wird.

```
bool PositionGetInteger (
    ENUM_POSITION_PROPERTY_INTEGER property_id,      // Eigenschaftsidentifikator
    long& long_var                                  // hier nehmen wir den Wert der Eigenschaft
);
```

### Parameter

*property\_id*

[in] Identifikator der Positionseigenschaft . Wert kann einer der Aufzählungswerte [ENUM\\_POSITION\\_PROPERTY\\_INTEGER](#) sein.

*long\_var*

[out] Variable der Art long, die den Wert der angeforderten Eigenschaft annimmt.

### Rückgabewert

Wert der Art [long](#). Im Fall der fehlerhaften Ausführung gibt 0 zurück.

### Bemerkung

Für jedes [Symbol](#) kann zu jedem Zeitpunkt nur eine [Position](#) geöffnet werden, die das Ergebnis eines oder mehrerer [Deals](#) ist. Man muss nicht Positionen und geltende [Warteordern](#) verwechseln, die auch in der Registerkarte "Handel" in der Werkzeugleiste "Instrumente" des Kundenterminals dargestellt werden.

Für bestimmte Erhaltung der frischen Daten über die Position ist es empfehlenswert, die Funktion [PositionSelect\(\)](#) aufzurufen, bevor man frische Daten aufruft.

### Beispiel:

```

//+-----+
//| Trade function |
//+-----+
void OnTrade()
{
//--- Überprüfen wir die Verfügbarkeit der Position und daraus Zeit ihrer Änderungen
    if(PositionSelect(_Symbol))
    {
//--- Indikator der Position für weitere Arbeit mit ihr erhalten
        ulong position_ID=PositionGetInteger(POSITION_IDENTIFIER);
        Print(_Symbol," postion #",position_ID);
//--- Zeit der Positionerstellung in Millisekunden seit 01.01.1970 erhalten
        long create_time_msc=PositionGetInteger(POSITION_TIME_MSC);
        PrintFormat("Position #%d POSITION_TIME_MSC = %i64 milliseconds => %s",position_ID,
            create_time_msc,TimeToString(create_time_msc/1000));
//--- Zeit der letzte Änderung der Position in Millisekunden seit 01.01.1970 erhalten
        long update_time_sec=PositionGetInteger(POSITION_TIME_UPDATE);
        PrintFormat("Position #%d POSITION_TIME_UPDATE = %i64 seconds => %s",
            position_ID,update_time_sec,TimeToString(update_time_sec));
//--- Zeit der letzte Änderung der Position in Millisekunden seit 01.01.1970 erhalten
        long update_time_msc=PositionGetInteger(POSITION_TIME_UPDATE_MSC);
        PrintFormat("Position #%d POSITION_TIME_UPDATE_MSC = %i64 milliseconds => %s",
            position_ID,update_time_msc,TimeToString(update_time_msc/1000));
    }
//---
}

```

Sehen Sie auch

[PositionGetSymbol\(\)](#), [PositionSelect\(\)](#), [Eigenschaften der Positionen](#)

## PositionGetString

Funktion gibt die angeforderte Eigenschaft der offenen Position zurück, die vorher für die Funktion [PositionGetSymbol](#) oder [PositionSelect](#) gewählt wurde. Eigenschaft der Position muss des Typs string sein. Es gibt 2 Varianten der Funktion.

1. Gibt den Wert der Eigenschaft sofort zurück.

```
string PositionGetString(  
    ENUM_POSITION_PROPERTY_STRING property_id    // Identifikator der Eigenschaft  
);
```

2. Gibt true oder false zurück abhängig davon, ob die Funktion erfolgreich durchgeführt wurde. Im Erfolgsfall wird der Wert der Eigenschaft in die Empfangsvariable gesetzt, die durch Referenz vom letzten Parameter übertragen wird.

```
bool PositionGetString(  
    ENUM_POSITION_PROPERTY_STRING property_id,    // Eigenschaftsidentifikator  
    string& string_var    // hier nehmen wir den Wert der Eigenschaft  
);
```

### Parameter

*property\_id*

[in] Identifikator der Positionseigenschaft. Wert kann einer der Aufzählungswerte [ENUM\\_POSITION\\_PROPERTY\\_STRING](#) sein.

*string\_var*

[out] Variable der Art string, die den Wert der angeforderten Eigenschaft annimmt.

### Rückgabewert

Wert der Art [string](#).

### Bemerkung

Für jedes [Symbol](#) kann zu jedem Zeitpunkt nur eine [Position](#) geöffnet werden, die das Ergebnis eines oder mehrerer [Deals](#) ist. Man muss nicht Positionen und geltende [Warteordern](#) verwechseln, die auch in der Registerkarte "Handel" in der Werkzeugleiste "Instrumente" des Kundenterminals dargestellt werden.

Für bestimmte Erhaltung der frischen Daten über die Position ist es empfehlenswert, die Funktion [PositionSelect\(\)](#) aufzurufen, bevor man frische Daten aufruft.

### Sehen Sie auch

[PositionGetSymbol\(\)](#), [PositionSelect\(\)](#), [Eigenschaften der Positionen](#)

## OrdersTotal

Gibt die Orderanzahl zurück.

```
int OrdersTotal();
```

### Rückgabewert

Wert der Art [int](#).

### Bemerkung

Man muss nicht geltende [Warteordern](#) und Positionen verwechseln, die auch in der Registerkarte "Handel" in der Werkzeugleiste "Instrumente" dargestellt werden. Order ist ein Befehl, eine [Handelsoperation](#) durchzuführen und Position ist das Ergebnis eines oder mehrerer [Deals](#).

Für jedes [Symbol](#) kann zu jedem Zeitpunkt nur eine [Position](#) geöffnet werden, während es mehrere Warteordern nach einem und demselben Symbol sein kann.

### Sehen Sie auch

[OrderSelect\(\)](#), [OrderGetTicket\(\)](#), [Ordereigenschaften](#)

## OrderGetTicket

Gibt Ticket der entsprechenden Order zurück und wählt automatisch Order für die weitere Arbeit damit mittels Funktionen .

```
ulong OrderGetTicket(
    int index // Nummer in Orderliste
);
```

### Parameter

*index*

[in] Ordernummer in Orderliste.

### Rückgabewert

Wert der Art [ulong](#).

### Bemerkung

Man muss nicht geltende [Warteordern](#) und Positionen verwechseln, die auch in der Registerkarte "Handel" in der Werkzeugleiste "Instrumente" dargestellt werden. Order ist ein Befehl, eine [Handelsoperation](#) durchzuführen, und Position ist das Ergebnis eines oder mehrerer [Deals](#).

Für jedes [Symbol](#) kann zu jedem Zeitpunkt nur eine [Position](#) geöffnet werden, während es mehrere Warteordern nach einem und demselben Symbol sein kann.

Funktion OrderGetTicket() kopiert Daten über die Order ins Programmumfeld und weitere Aufrufe [OrderGetDouble\(\)](#), [OrderGetInteger\(\)](#), [OrderGetString\(\)](#) geben die früher kopierten Daten zurück. Das bedeutet, dass die Order selbst nicht mehr sein (oder darin haben sich Eröffnungspreis, Levels Stop Loss / Take Profit oder Gültigkeitsfrist verändert), aber die Daten nach dieser Order können noch erhalten werden. Für bestimmte Erhaltung der frischen Daten über die Position ist es empfehlenswert, die Funktion OrderGetTicket() aufzurufen, bevor man frische Daten aufruft

### Beispiel:

```
void OnStart()
{
    //--- Variablen für Erhaltung der Werte aus Ordereigenschaften
    ulong   ticket;
    double  open_price;
    double  initial_volume;
    datetime time_setup;
    string   symbol;
    string   type;
    long     order_magic;
    long     positionID;
    //--- Anzahl der laufenden Warteorder
    uint     total=OrdersTotal();
    //--- gehen wir im Zyklus durch alle Ordnern
    for(uint i=0;i<total;i++)
    {
        //--- erhalten wir Orderticket in Bezug auf seine Position in der Liste
```

```

if((ticket=OrderGetTicket(i))>0)
{
    //--- erhlaten wir Ordereigenschaften
    open_price    =OrderGetDouble(ORDER_PRICE_OPEN);
    time_setup    =(datetime)OrderGetInteger(ORDER_TIME_SETUP);
    symbol        =OrderGetString(ORDER_SYMBOL);
    order_magic   =OrderGetInteger(ORDER_MAGIC);
    positionID    =OrderGetInteger(ORDER_POSITION_ID);
    initial_volume=OrderGetDouble(ORDER_VOLUME_INITIAL);
    type          =EnumToString(ENUM_ORDER_TYPE(OrderGetInteger(ORDER_TYPE)));
    //--- bereiten wir die Information über Order vor und geben sie aus
    printf("#ticket %d %s %G %s at %G was set up at %s",
        ticket,                // Orderticket
        type,                  // Typ
        initial_volume,        // das gestellte Volumen
        symbol,                // Symbol, für das gestellt ist
        open_price,            // der angegebene Eröffnungspreis
        TimeToString(time_setup)// Zeit der Orderstellung
    );
};
};
//---
};

```

Sehen Sie auch

[OrdersTotal\(\)](#), [OrderSelect\(\)](#), [OrderGetInteger\(\)](#)

## OrderSelect

Wählt Order für die weitere Arbeit damit. Gibt true bei erfolgreicher Funktionsausführung zurück. Gibt false bei verfehlter Funktionsausführung zurück. Um fehlerbezogene Information zu bekommen, muss die Funktion [GetLastError\(\)](#) gehiessen werden.

```
bool OrderSelect (
    ulong   ticket,      // Orderticket
);
```

### Parameter

*ticket*

[in] Orderticket.

### Rückgabewert

Wert der Art bool.

### Bemerkung

Man muss nicht geltende [Warteordern](#) und Positionen verwechseln, die auch in der Registerkarte "Handel" in der Werkzeugleiste "Instrumente" des Kundenterminals dargestellt werden. Für jedes [Symbol](#) kann zu jedem Zeitpunkt nur eine [Position](#) geöffnet werden, während es mehrere Warteordern nach einem und demselben Symbol sein kann.

Funktion OrderSelect() kopiert Daten über die Order ins Programmumfeld und weitere Aufrufe [OrderGetDouble\(\)](#), [OrderGetInteger\(\)](#), [OrderGetString\(\)](#) geben die früher kopierten Daten zurück. Das bedeutet, dass die Order selbst nicht mehr sein kann (oder darin haben sich Eröffnungspreis, Levels Stop Loss / Take Profit oder Gültigkeitsfrist verändert), aber Daten nach dieser Order können noch erhalten werden. Für bestimmte Erhaltung der frischen Daten über die Position ist es empfehlenswert, die Funktion OrderSelect() aufzurufen, bevor man frische Daten aufruft.

### Sehen Sie auch

[OrderGetInteger\(\)](#), [OrderGetDouble\(\)](#), [OrderGetString\(\)](#), [OrderCalcProfit\(\)](#), [OrderGetTicket\(\)](#), [Ordereigenschaften](#)



## OrderGetDouble

Gibt die angeforderte Ordereigenschaft zurück, vorläufig gewählt mittels der Funktionen [OrderGetTicket](#) oder [OrderSelect](#). Ordereigenschaft muss der Art double. Es gibt zwei Arten der Funktion.

1. Gibt unmittelbar die Eigenschaftsgroesse zurück.

```
double OrderGetDouble(
    ENUM_ORDER_PROPERTY_DOUBLE property_id // Eigenschaftsidentifikator
);
```

2. Gibt true oder false zurück, abhängig davon, ob die Funktion erfolgreich durchgeführt wird. Im Erfolgsfall wird der Eigenschaftswert in eine Empfangsvariable gestellt, die vom letzten Parameter durch Referenz übertragen wird.

```
bool OrderGetDouble(
    ENUM_ORDER_PROPERTY_DOUBLE property_id, // Eigenschaftsidentifikator
    double& double_var // hier erhalten wir den Wert der Eigenschaft
);
```

### Parameter

*property\_id*

[in] Ordereigenschaftsidentifikator. Die Größe kann eine der Abzählungsgroessen [ENUM\\_ORDER\\_PROPERTY\\_DOUBLE](#) sein.

*double\_var*

[out] Variable der Art double, die den Wert der angeforderten Eigenschaft annimmt.

### Rückgabewert

Wert der Art [double](#).

### Bemerkung

Man muss nicht geltende [Warteordern](#) und Positionen verwechseln, die auch in der Registerkarte "Handel" in der Werkzeugleiste "Instrumente" des Kundenterminals dargestellt werden. Für jedes [Symbol](#) kann zu jedem Zeitpunkt nur eine [Position](#) geöffnet werden, während es mehrere Warteordern nach einem und demselben Symbol sein kann.

Für bestimmte Erhaltung der frischen Daten über die Position ist es empfehlenswert, die Funktion [OrderSelect\(\)](#) aufzurufen, bevor man frische Daten aufruft.

### Sehen Sie auch

[OrdersTotal\(\)](#), [OrderGetTicket\(\)](#), [Ordereigenschaften](#)

## OrderGetInteger

Gibt die angeforderte Eigenschaft der Order zurück, die vorher durch die Funktion [OrderGetTicket](#) oder [OrderSelect](#) gewählt wurde. Eigenschaft der Order muss des Typs datetime, int sein. Es gibt 2 Varianten der Funktion.

1. Gibt den Wert der Eigenschaft sofort zurück.

```
long OrderGetInteger(
    ENUM_ORDER_PROPERTY_INTEGER property_id // Identifikator der Eigenschaft
);
```

2. Gibt true oder false abhängig davon, ob die Funktion erfolgreich durchgeführt wurde. Im Erfolgsfall wird der Wert der Eigenschaft in die Empfangsvariable gestellt, die durch Referenz vom letzten Parameter übertragen wird.

```
bool OrderGetInteger(
    ENUM_ORDER_PROPERTY_INTEGER property_id, // Identifikator der Eigenschaft
    long& long_var // hier nehmen wir den Wert der Eigenschaft
);
```

### Parameter

*property\_id*

[in] Identifikator der Ordereigenschaft. Wert kann einer der Aufzählungswerte [ENUM\\_ORDER\\_PROPERTY\\_INTEGER](#) sein.

*long\_var*

[out] Variable des Typs long, die den Wert der angeforderten Eigenschaft annimmt.

### Rückgabewert

Wert des Typs [long](#).

### Bemerkung

Man muss nicht geltende [Warteordern](#) und Positionen verwechseln, die in der Registerkarte "Handel" in der Werkzeugleiste "Instrumente" des Kundenterminals dargestellt werden. Für jedes [Symbol](#) kann zu jedem Zeitpunkt nur eine [Position](#) geöffnet werden, während es mehrere Warteordern nach einem und demselben Symbol sein kann.

Für bestimmte Erhaltung der frischen Daten über die Order ist es empfehlenswert, die Funktion [OrderSelect\(\)](#) aufzurufen, bevor man frische Daten aufruft.

### Sehen Sie auch

[OrdersTotal\(\)](#), [OrderGetTicket\(\)](#), [Ordereigenschaften](#)

## OrderGetString

Gibt die angeforderte Ordereigenschaft, die vorher durch die Funktion [OrderGetTicket](#) oder [OrderSelect](#) gewählt wurde. Eigenschaft der Order muss des Typs string sein. Es gibt 2 Varianten der Funktion.

1. Gibt den Wert der Eigenschaft sofort zurück.

```
string OrderGetString(  
    ENUM_ORDER_PROPERTY_STRING property_id // Identifikator der Eigenschaft  
);
```

2. Gibt true oder false abhängig davon, ob die Funktion erfolgreich durchgeführt wurde. Im Erfolgsfall wird der Wert der Eigenschaft in die Empfangsvariable gestellt, die durch Referenz vom letzten Parameter übertragen wird.

```
bool OrderGetString(  
    ENUM_ORDER_PROPERTY_STRING property_id, // Identifikator der Eigenschaft  
    string& string_var // hier nehmen wir den Wert der Eigenschaft  
);
```

### Parameter

*property\_id*

[in] Identifikator der Ordereigenschaft. Wert kann einer der Aufzählungswerte [ENUM\\_ORDER\\_PROPERTY\\_STRING](#) sein.

*string\_var*

[out] Variable des Typs string, die den Wert der angeforderten Eigenschaft annimmt.

### Rückgabewert

Wert des Typs [string](#).

### Bemerkung

Man muss nicht geltende [Warteordern](#) und Positionen verwechseln, die in der Registerkarte "Handel" in der Werkzeugleiste "Instrumente" des Kundenterminals dargestellt werden. Für jedes [Symbol](#) kann zu jedem Zeitpunkt nur eine [Position](#) geöffnet werden, während es mehrere Warteordern nach einem und demselben Symbol sein kann.

Für bestimmte Erhaltung der frischen Daten über die Order ist es empfehlenswert, die Funktion [OrderSelect\(\)](#) aufzurufen, bevor man frische Daten aufruft.

### Sehen Sie auch

[OrdersTotal\(\)](#), [OrderGetTicket\(\)](#), [Ordereigenschaften](#)

## HistorySelect

Ruft die Geschichte der Deals und Ordnern für die angegebene Periode der Serverzeit ab.

```
bool HistorySelect(
    datetime from_date,    // von Datum
    datetime to_date       // bis Datum
);
```

### Parameter

*from\_date*

[in] Anfangsdatum der Anforderung.

*to\_date*

[in] Beendigungsdatum der Anforderung.

### Rückgabewert

Gibt true im Erfolgsfall zurück, anderenfalls false.

### Bemerkung

Funktion HistorySelect() erzeugt im mql5-Programm die Liste der Ordnern und die Liste der Deals für weitere Aufrufe der Elemente der Liste mittels der entsprechenden Funktionen. Die Größe der Liste der Deals kann man mittels der Funktion [HistoryDealsTotal\(\)](#) erfahren, die Größe der Liste der Ordnern in der Geschichte kann man mittels [HistoryOrdersTotal\(\)](#) erfahren. Elemente der Liste der Ordnern ist es am besten mittels der Funktion [HistoryOrderGetTicket\(\)](#) durchzusehen, für Elemente der Liste der Deals passt die Funktion [HistoryDealGetTicket\(\)](#).

Nach Verwendung der Funktion [HistoryOrderSelect\(\)](#) wird die Liste der Ordnern in der Geschichte, die für mql5-Programm zugänglich ist, gelöscht und von einer erneut gefundenen Order ausgefüllt, wenn die [Suche der Order nach Ticket](#) erfolgreich beendet hat. Dasselbe bezieht sich auf die Liste der Deals, die für mql5-Programm zugänglich ist - sie wird von der Funktion [HistoryDealSelect\(\)](#) gelöscht und erneut ausgefüllt, wenn der Deal nach Ticketnummer erfolgreich erhalten ist.

### Beispiel:

```
void OnStart()
{
    color BuyColor = clrBlue;
    color SellColor = clrRed;
    ///--- request trade history
    HistorySelect(0, TimeCurrent());
    ///--- create objects
    string name;
    uint total = HistoryDealsTotal();
    ulong ticket = 0;
    double price;
    double profit;
    datetime time;
    string symbol;
    long type;
```

```

    long    entry;
    //--- for all deals
    for(uint i=0;i<total;i++)
    {
        //--- try to get deals ticket
        if((ticket=HistoryDealGetTicket(i))>0)
        {
            //--- get deals properties
            price =HistoryDealGetDouble(ticket,DEAL_PRICE);
            time  =(datetime)HistoryDealGetInteger(ticket,DEAL_TIME);
            symbol=HistoryDealGetString(ticket,DEAL_SYMBOL);
            type   =HistoryDealGetInteger(ticket,DEAL_TYPE);
            entry  =HistoryDealGetInteger(ticket,DEAL_ENTRY);
            profit=HistoryDealGetDouble(ticket,DEAL_PROFIT);
            //--- only for current symbol
            if(price && time && symbol==Symbol())
            {
                //--- create price object
                name="TradeHistory_Deal_"+string(ticket);
                if(entry) ObjectCreate(0,name,OBJ_ARROW_RIGHT_PRICE,0,time,price,0,0);
                else      ObjectCreate(0,name,OBJ_ARROW_LEFT_PRICE,0,time,price,0,0);
                //--- set object properties
                ObjectSetInteger(0,name,OBJPROP_SELECTABLE,0);
                ObjectSetInteger(0,name,OBJPROP_BACK,0);
                ObjectSetInteger(0,name,OBJPROP_COLOR,type?BuyColor:SellColor);
                if(profit!=0) ObjectSetString(0,name,OBJPROP_TEXT,"Profit: "+string(profit));
            }
        }
    }
    //--- apply on chart
    ChartRedraw();
}

```

### Sehen Sie auch

[HistoryOrderSelect\(\)](#), [HistoryDealSelect\(\)](#)

## HistorySelectByPosition

Fordert die Geschichte der Deals und der Order mit dem angegebenen [Identifikator der Position](#) an.

```
bool HistorySelectByPosition(  
    long position_id // Identifikator der Position - POSITION\_IDENTIFIER  
);
```

### Parameter

*position\_id*

[in] Der Identifikator der Position, der auf jede ausgeführte Auftrag und auf jede Transaktion festgelegt ist.

### Rückgabewert

Gibt true im Erfolgsfall zurück, anderenfalls false.

### Bemerkung

Man muss nicht Ordern aus der Handelsgeschichte und geltende [Warteordern](#) verwechseln, die in der Registerkarte "Handel" in der Werkzeugleiste "Instrumente" dargestellt werden. Liste der [Ordern](#), die verändert wurden oder zur Ausführung der Handelsoperation geführt haben, kann man in der Registerkarte "Geschichte" in der Werkzeugleiste "Instrumente" des Kundenterminals sehen.

Die Funktion HistorySelectByPosition() erzeugt im mql5-Programm die Liste der Ordern und die Liste der Deals mit dem angegebenen [Identifikator der Position](#) für weitere Aufrufe der Elemente der Liste mittels der entsprechenden Funktionen. Die Grösse der Liste kann man mittels der Funktion [HistoryDealsTotal\(\)](#) erfahren, die Größe der Liste der Ordern kann man mittels [HistoryOrdersTotal\(\)](#) erfahren. Elemente der Liste der Ordern ist es besser, mittels der Funktion [HistoryOrderGetTicket\(\)](#) durchzusehen, für Elemente der Liste der Deals passt am besten die Funktion [HistoryDealGetTicket\(\)](#).

Nach der Verwendung der Funktion [HistoryOrderSelect\(\)](#) wird die Liste der Ordern in der Geschichte, die für mql5-Programm zugänglich ist, gelöscht und von einer erneut gefundenen Order ausgefüllt, wenn die [Suche der Order nach Ticket](#) erfolgreich beendet hat. Dasselbe bezieht sich auf die Liste der Deals, die für mql5-Programm zugänglich ist - sie wird von der Funktion [HistoryDealSelect\(\)](#) gelöscht und erneut ausgefüllt, wenn der Deal nach Ticketnummer erfolgreich erhalten ist.

### Sehen Sie auch

[HistorySelect\(\)](#), [HistoryOrderGetTicket\(\)](#), [Ordereigenschaften](#)

## HistoryOrderSelect

Wählt in der Geschichte Order für den weiteren Zugang dazu durch die entsprechenden Funktionen. Gibt true zurück, wenn die Durchführung der Funktion erfolgreich beendet wird. Gibt false zurück wenn die Durchführung der Funktion erfolglos beendet wird. Für die Erhaltung der fehlerbezogenen Information rufen Sie die Funktion [GetLastError\(\)](#) auf.

```
bool HistoryOrderSelect(  
    ulong ticket,      // Orderticket  
);
```

### Parameter

*ticket*

[in] Orderticket

### Rückgabewert

Gibt true im Erfolgsfall zurück, anderenfalls false.

### Bemerkung

Man muss nicht Order aus der Handelsgeschichte und geltende [Warteordern](#) verwechseln, die in der Registerkarte "Handel" in der Werkzeugleiste "Instrumente" dargestellt werden. Die Liste der [Ordern](#), die annulliert wurden oder zur Ausführung der Handelsoperation geführt haben, kann man in der Registerkarte "Geschichte" in der Werkzeugleiste "Instrumente" des Kundenterminals sehen.

Funktion HistoryOrderSelect() löscht im mql5-Programm die Liste der Order aus der Geschichte, die für Aufrufe zugänglich sind, und kopiert darin die einzige Order, wenn die Ausführung HistoryOrderSelect() erfolgreich beendet hat. Wenn es notwendig ist, alle Deals durchzusehen, die von der Funktion [HistorySelect\(\)](#) gewählt wurden, ist es besser die Funktion [HistoryOrderGetTicket\(\)](#) zu verwenden.

### Sehen Sie auch

[HistorySelect\(\)](#), [HistoryOrderGetTicket\(\)](#), [Ordereigenschaften](#)

## HistoryOrdersTotal

Gibt die Anzahl der Order in der Geschichte zurück. Vor dem Aufruf von der Funktion `HistoryOrdersTotal()`, müssen Sie die Geschichte der Transaktionen und Aufträge mit der [HistorySelect\(\)](#) oder [HistorySelectByPosition\(\)](#) erhalten.

```
int HistoryOrdersTotal();
```

### Rückgabewert

Wert des Typs [int](#).

### Bemerkung

Man muss nicht Ordern aus der Handelsgeschichte und geltende [Warteordern](#) verwechseln, die in der Registerkarte "Handel" in der Werkzeugleiste "Instrumente" dargestellt werden. Die Liste der [Ordern](#), die annulliert wurden oder zur Ausführung der Handelsoperation geführt haben, kann man in der Registerkarte "Geschichte" in der Werkzeugleiste "Instrumente" des Kundenterminals sehen.

### Sehen Sie auch

[HistorySelect\(\)](#), [HistoryOrderSelect\(\)](#), [HistoryOrderGetTicket\(\)](#), [Ordereigenschaften](#)



## HistoryOrderGetTicket

Gibt Ticket der entsprechenden Order in der Geschichte zurück. Prior to calling HistoryOrderGetTicket(), first it is necessary to receive the history of deals and orders using the [HistorySelect\(\)](#) or [HistorySelectByPosition\(\)](#)

```
ulong HistoryOrderGetTicket (
    int index      // Nummer in der Orderliste
);
```

### Parameter

*index*

[in] Ordernummer in der Orderliste.

### Rückgabewert

Wert des Typs [ulong](#). Im Fall der fehlerhaften Ausführung gibt 0 zurück.

### Bemerkung

Man muss nicht Ordern aus der Handelsgeschichte und geltende [Warteordern](#) verwechseln, die in der Registerkarte "Handel" in der Werkzeugleiste "Instrumente" dargestellt werden. Die Liste der [Ordern](#), die annulliert wurden oder zur Ausführung der Handelsoperation geführt haben, kann man in der Registerkarte "Geschichte" in der Werkzeugleiste "Instrumente" des Kundenterminals sehen.

### Beispiel:

```
void OnStart ()
{
    datetime from=0;
    datetime to=TimeCurrent();
    //--- fordern wir die ganze Geschichte an
    HistorySelect(from,to);
    //--- Variablen für Erhaltung der Werte aus Ordereigenschaften
    ulong    ticket;
    double   open_price;
    double   initial_volume;
    datetime time_setup;
    datetime time_done;
    string    symbol;
    string    type;
    long     order_magic;
    long     positionID;
    //--- Anzahl der laufenden suspendierten Order
    uint     total=HistoryOrdersTotal();
    //--- Gehen wir im Zyklus durch alle Order
    for(uint i=0;i<total;i++)
    {
        //--- Erhalten wir Ticket der Order nach ihrer Position in der Liste
        if((ticket=HistoryOrderGetTicket(i))>0)
        {
```

```

    ///--- Erhalten wir Eigenschaften der Order
    open_price=      HistoryOrderGetDouble(ticket,ORDER_PRICE_OPEN);
    time_setup=      (datetime)HistoryOrderGetInteger(ticket,ORDER_TIME_SETUP);
    time_done=       (datetime)HistoryOrderGetInteger(ticket,ORDER_TIME_DONE);
    symbol=          HistoryOrderGetString(ticket,ORDER_SYMBOL);
    order_magic=     HistoryOrderGetInteger(ticket,ORDER_MAGIC);
    positionID =     HistoryOrderGetInteger(ticket,ORDER_POSITION_ID);
    initial_volume=  HistoryOrderGetDouble(ticket,ORDER_VOLUME_INITIAL);
    type=GetOrderType(HistoryOrderGetInteger(ticket,ORDER_TYPE));
    ///--- bereiten wir die information über Order vor und geben sie aus
    printf("#ticket %d %s %G %s at %G was set up at %s => done at %s, pos ID=%d",
        ticket,          // Orderticket
        type,            // Typ
        initial_volume,  // das gestellte Volumen
        symbol,          // Symbol, für das gestellt wird
        open_price,      // der angegeben Eroeffnungspreis
        TimeToString(time_setup), // Zeit der Ordereinstellung
        TimeToString(time_done), // Zeit der Durchführung oder der Entfernung
        positionID       // ID der Position, in die das Ordergeschaefte
    );

    ;}

; }
///---
; }
//+-----+
//| Gibt die Zeilenbenennung des Ordertyps zurück |
//+-----+
string GetOrderType(long type)
{
    string str_type="unknown operation";
    switch(type)
    {
        case (ORDER_TYPE_BUY):          return("buy");
        case (ORDER_TYPE_SELL):         return("sell");
        case (ORDER_TYPE_BUY_LIMIT):    return("buy limit");
        case (ORDER_TYPE_SELL_LIMIT):   return("sell limit");
        case (ORDER_TYPE_BUY_STOP):     return("buy stop");
        case (ORDER_TYPE_SELL_STOP):    return("sell stop");
        case (ORDER_TYPE_BUY_STOP_LIMIT): return("buy stop limit");
        case (ORDER_TYPE_SELL_STOP_LIMIT): return("sell stop limit");
    }
    ;}
    return(str_type);
; }

```

Sehen Sie auch

[HistorySelect\(\)](#), [HistoryOrdersTotal\(\)](#), [HistoryOrderSelect\(\)](#), [Ordereigenschaften](#)

## HistoryOrderGetDouble

Gibt die angeforderte Eigenschaft der Order zurück. Eigenschaft der Order muss des Typs double sein. Es gibt 2 Varianten der Funktion.

1. Gibt den Wert der Eigenschaft sofort zurück.

```
double HistoryOrderGetDouble(
    ulong          ticket_number,    // Ticket
    ENUM_ORDER_PROPERTY_DOUBLE property_id // Identifikator der Eigenschaft
);
```

2. Gibt true oder false zurück, abhängig von der erfolgreichen Durchführung der Funktion. Im Erfolgsfall wird Wert der Eigenschaft in die Empfangsvariable gesetzt, die durch Referenz vom letzten Parameter übertragen wird.

```
bool HistoryOrderGetDouble(
    ulong          ticket_number,    // Ticket
    ENUM_ORDER_PROPERTY_DOUBLE property_id, // Identifikator der Eigenschaft
    double&        double_var       // hier nehmen wir den Wert der Eigenschaft
);
```

### Parameter

*ticket\_number*

[in] Orderticket.

*property\_id*

[in] Identifikator der Ordereigenschaft. Wert kann einer der Aufzählungswerte [ENUM\\_ORDER\\_PROPERTY\\_DOUBLE](#) sein.

*double\_var*

[out] Variable des Typs double, die den Wert der angeforderten Eigenschaft annimmt.

### Rückgabewert

Wert des Typs [double](#).

### Bemerkung

Man muss nicht Ordern aus der Handelsgeschichte und geltende [Warteordern](#) verwechseln, die in der Registerkarte "Handel" in der Werkzeugleiste "Instrumente" sehen. Liste der [Ordern](#), die annulliert wurden oder zur Ausführung der Handelsoperation geführt haben, kann man in der Registerkarte "Geschichte" in der Werkzeugleiste "Instrumente" des Kundenterminals sehen.

### Sehen Sie auch

[HistorySelect\(\)](#), [HistoryOrdersTotal\(\)](#), [HistoryOrderSelect\(\)](#), [Ordereigenschaften](#)

## HistoryOrderGetInteger

Gibt die angeforderte Eigenschaft der Order. Eigenschaft der Order muss des Typs datetime, int sein. Es gibt 2 Varianten der Funktion.

1. Gibt den Wert der Eigenschaft sofort zurück.

```
long HistoryOrderGetInteger(
    ulong          ticket_number,    // Ticket
    ENUM_ORDER_PROPERTY_INTEGER property_id // Identifikator der Eigenschaft
);
```

2. Gibt true oder false zurück abhängig davon, ob die Funktion erfolgreich durchgeführt wurde. Im Erfolgsfall wird Wert der Eigenschaft in die Empfangsvariable gesetzt, die durch Referenz vom letzten Parameter übertragen wird.

```
bool HistoryOrderGetInteger(
    ulong          ticket_number,    // Ticket
    ENUM_ORDER_PROPERTY_INTEGER property_id, // Identifikator der Eigenschaft
    long&          long_var         // hier nehmen wir den Wert der Eigenschaft
);
```

### Parameter

*ticket\_number*

[in] Ticket der Order.

*property\_id*

[in] Identifikator der Ordereigenschaft. Wert kann einer der Aufzählungswerte [ENUM\\_ORDER\\_PROPERTY\\_INTEGER](#).

*long\_var*

[out] Variable des Typs long, die den Wert der angeforderten Eigenschaft annimmt.

### Rückgabewert

Wert des Typs [long](#).

### Bemerkung

Man muss nicht Ordern aus der Handelsgeschichte und geltende [Warteordern](#) verwechseln, die in der Registerkarte "Handel" in der Werkzeugleiste "Instrumente" dargestellt werden. Liste der [Ordern](#), die annulliert wurden oder zur Ausführung der Handelsoperation geführt haben, kann man in der Registerkarte "Geschichte" in der Werkzeugleiste "Instrumente" des Kundenterminals sehen.

### Beispiel:

```

//+-----+
//| Trade function                                     |
//+-----+
void OnTrade()
{
//--- Ticket des letzten Order aus der Handelsgeschichte für eine Woche erhalten
ulong last_order=GetLastOrderTicket();
if(HistoryOrderSelect(last_order))
{
//--- Ordersetzungszeit in Millisekunden seit 01.01.1970
long time_setup_msc=HistoryOrderGetInteger(last_order,ORDER_TIME_SETUP_MSC);
PrintFormat("Order #d ORDER_TIME_SETUP_MSC=%i64 => %s",
            last_order,time_setup_msc,TimeToString(time_setup_msc/1000));
//--- Orderausführung- oder Löschungszeit in Millisekunden seit 01.01.1970
long time_done_msc=HistoryOrderGetInteger(last_order,ORDER_TIME_DONE_MSC);
PrintFormat("Order #d ORDER_TIME_DONE_MSC=%i64 => %s",
            last_order,time_done_msc,TimeToString(time_done_msc/1000));
}
else // Den Fehler melden
    PrintFormat("HistoryOrderSelect() failed for #d. Error code=%d",
                last_order,GetLastError());

//---
}
//+-----+
//| Gibt den Ticket des letzten Orders in der Geschichte oder -1 |
//+-----+
ulong GetLastOrderTicket()
{
//--- Geschichte für die letzte 7 Tagen anfragen
if(!GetTradeHistory(7))
{
//--- Nachricht über fehlgeschlagenen Aufruf und Rückgabe -1
Print(__FUNCTION__," HistorySelect() hat false zurückgegeben");
return -1;
}
//---
ulong first_order,last_order,orders=HistoryOrdersTotal();
//--- wenn es Ordern gibt, anfangen wir mit ihnen zu arbeiten
if(orders>0)
{
    Print("Orders = ",orders);
    first_order=HistoryOrderGetTicket(0);
    PrintFormat("first_order = %d",first_order);
    if(orders>1)
    {
        last_order=HistoryOrderGetTicket((int)orders-1);
        PrintFormat("last_order = %d",last_order);
        return last_order;
    }
    return first_order;
}
//--- keine Orders waren gefunden, Rückgabe -1
return -1;
}
//+-----+
//| Geschichte für letzte Tage anfragen, gibt false bei Fehler zurück|
//+-----+
bool GetTradeHistory(int days)
{
//--- definieren wir Wochenperiode, um eine Handelsgeschichte zu beantragen

```

```
datetime to=TimeCurrent();
datetime from=to-days*PeriodSeconds(PERIOD_D1);
ResetLastError();
//--- Machen wir eine Anfrage und überprüfen wir die Ergebnisse
if(!HistorySelect(from,to))
{
    Print(__FUNCTION__," HistorySelect=false. Error code=",GetLastError());
    return false;
}
//--- Geschichte erfolgreich empfangen
return true;
}
```

**Sehen Sie auch**

[HistorySelect\(\)](#), [HistoryOrdersTotal\(\)](#), [HistoryOrderSelect\(\)](#), [Ordereigenschaften](#)

## HistoryOrderGetString

Gibt die angeforderte Eigenschaft der Order.Ordereigenschaft muss des Typs string sein. Es gibt 2 Varianten der Funktion.

1. Gibt den Wert der Eigenschaft sofort zurück.

```
string HistoryOrderGetString(
    ulong          ticket_number,    // Ticket
    ENUM_ORDER_PROPERTY_STRING property_id // Identifikator der Eigenschaft
);
```

2. Gibt true oder false zurück, abhängig von der erfolgreichen Durchführung der Funktion. Im Erfolgsfall wird der Wert der Eigenschaft in die Empfangsvariable gesetzt, die durch Referenz vom letzten Parameter übertragen wird.

```
bool HistoryOrderGetString(
    ulong          ticket_number,    // Ticket
    ENUM_ORDER_PROPERTY_STRING property_id, // Identifikator der Eigenschaft
    string&        string_var       // hier nehmen wir den Wert der Eigenschaft
);
```

### Parameter

*ticket\_number*

[in] Orderticket.

*property\_id*

[in] Identifikator der Ordereigenschaft. Wert kann einer der Aufzählungswerte [ENUM\\_ORDER\\_PROPERTY\\_STRING](#) sein.

*string\_var*

[out] Variable des Typs string, die den Wert der angeforderten Eigenschaft annimmt.

### Rückgabewert

Wert des Typs [string](#).

### Bemerkung

Man muss nicht Ordern aus der Handelsgeschichte und geltende [Warteordern](#) verwechseln, die in der Registerkarte "Handel" in der Werkzeugleiste "Instrumente" dargestellt werden. Liste der [Ordern](#), die annulliert wurden oder zur Ausführung der Handelsoperation geführt haben, kann man in der Registerkarte "Geschichte" in der Werkzeugleiste "Instrumente" des Kundenterminals sehen.

### Sehen Sie auch

[HistorySelect\(\)](#), [HistoryOrdersTotal\(\)](#), [HistoryOrderSelect\(\)](#), [Ordereigenschaften](#)

## HistoryDealSelect

Wählt den Deal in der Geschichte für den weiteren Zugang dazu durch die entsprechenden Funktionen. Gibt true beim erfolgreichen Beenden der Funktion. Gibt false beim erfolglosen Beenden der Funktion zurück. Für die Erhaltung der fehlerbezogenen Information muss die Funktion [GetLastError\(\)](#) aufgerufen werden.

```
bool HistoryDealSelect(  
    ulong ticket,    // Dealsticket  
);
```

### Parameter

*ticket*

[in] Dealsticket

### Rückgabewert

Gibt true im Erfolgsfall zurück, anderenfalls false.

### Bemerkung

Man muss nicht [Ordern](#), [Deals](#) und [Positionen](#) verwechseln. Jeder Deal ist das Ergebnis der Ausführung einer Order, jede Position ist das Abschlussergebnis eines oder mehrerer Deals.

Funktion HistoryDealSelect() löscht im mql5-Programm die Liste der Deals, die für Aufrufe zugänglich sind, und kopiert darin der einzige Deal, wenn die Ausführung HistoryDealSelect() erfolgreich beendet hat. Wenn es notwendig ist, alle Deals durchzusehen, die von der Funktion [HistorySelect\(\)](#) gewählt wurden, ist es besser die Funktion [HistoryDealGetTicket\(\)](#) zu verwenden.

### Sehen Sie auch

[HistorySelect\(\)](#), [HistoryDealGetTicket\(\)](#), [Eigenschaften der Deals](#)



## HistoryDealsTotal

Gibt die Anzahl der Deals in der Geschichte zurück. Vor dem Aufruf von der Funktion HistoryDealsTotal(), müssen Sie die Geschichte der Transaktionen und Aufträge mit der [HistorySelect\(\)](#) oder [HistorySelectByPosition\(\)](#) erhalten.

```
int HistoryDealsTotal();
```

### Rückgabewert

Wert des Typs [int](#).

### Bemerkung

Man muss nicht [Ordern](#), [Deals](#) und [Positionen](#) verwechseln. Jedes Geschäft ist das Ergebnis der Ausführung einer Order, jede Position ist das Abschlussergebnis eines oder mehrerer Deals.

### Sehen Sie auch

[HistorySelect\(\)](#), [HistoryDealGetTicket\(\)](#), [Eigenschaften der Deals](#)

## HistoryDealGetTicket

Wählt den Deal für die weitere Verarbeitung und gibt Ticket des Deales in der Geschichte zurück. Prior to calling HistoryDealGetTicket(), first it is necessary to receive the history of deals and orders using the [HistorySelect\(\)](#) or [HistorySelectByPosition\(\)](#) function.

```
ulong HistoryDealGetTicket (
    int index    // Nummer des Deals
);
```

### Parameter

*index*

[in] Nummer des Deals in der Dealsliste.

### Rückgabewert

Wert des Typs [ulong](#). Im Fall der fehlerhaften Ausführung gibt 0 zurück.

### Bemerkung

Man muss nicht [Ordern](#), [Deals](#) und [Positionen](#) verwechseln. Jeder Deal ist das Ergebnis der Ausführung einer Order, jede Position ist das Abschlussergebnis eines oder mehrerer Deals.

### Beispiel:

```

void OnStart()
{
    ulong deal_ticket;           // Ticket des Deals
    ulong order_ticket;         // Ticket der Order, für den Deal durchgeführt wurde
    datetime transaction_time;   // Zeit der Transaktion
    long deal_type;             // Typ der Handelsoperation
    long position_ID;           // Identifikator der Position
    string deal_description;     // Beschreibung der Operation
    double volume;              // Volumen der Operation
    string symbol;              // Symbol des Deals
    //--- stellen wir das Anfangsdatum und das Schlussdatum für Anforderung der Dealssges
    datetime from_date=0;       // vom Anfang
    datetime to_date=TimeCurrent(); // bis zum laufenden Zeitpunkt
    //--- fordern wir die Dealsgeschichte im angegebenen Intervall an
    HistorySelect(from_date,to_date);
    //--- gesamte Anzahl in der Liste der Deals
    int deals=HistoryDealsTotal();
    //--- jetzt verarbeiten wir jeden Deal
    for(int i=0;i<deals;i++)
    {
        deal_ticket=             HistoryDealGetTicket(i);
        volume=                  HistoryDealGetDouble(deal_ticket,DEAL_VOLUME);
        transaction_time=(datetime)HistoryDealGetInteger(deal_ticket,DEAL_TIME);
        order_ticket=            HistoryDealGetInteger(deal_ticket,DEAL_ORDER);
        deal_type=                HistoryDealGetInteger(deal_ticket,DEAL_TYPE);
        symbol=                   HistoryDealGetString(deal_ticket,DEAL_SYMBOL);
        position_ID=              HistoryDealGetInteger(deal_ticket,DEAL_POSITION_ID);
        deal_description=         GetDealDescription(deal_type,volume,symbol,order_tic
        //--- machen wir das schoene Formatieren für Dealnummer
        string print_index=StringFormat("% 3d",i);
        //--- geben wir Information über den Deal aus
        Print(print_index+": deal #",deal_ticket," at ",transaction_time,deal_descriptio
    }
}

//+-----+
//| Gibt die Zeilenbeschreibung der Operation zurück |
//+-----+
string GetDealDescription(long deal_type,double volume,string symbol,long ticket,long
{
    string descr;
    //---
    switch(deal_type)
    {

```

```

        case DEAL_TYPE_BALANCE:           return ("balance");
        case DEAL_TYPE_CREDIT:            return ("credit");
        case DEAL_TYPE_CHARGE:            return ("charge");
        case DEAL_TYPE_CORRECTION:        return ("correction");
        case DEAL_TYPE_BUY:               descr="buy"; break;
        case DEAL_TYPE_SELL:              descr="sell"; break;
        case DEAL_TYPE_BONUS:             return ("bonus");
        case DEAL_TYPE_COMMISSION:        return ("additional commission");
        case DEAL_TYPE_COMMISSION_DAILY:  return ("daily commission");
        case DEAL_TYPE_COMMISSION_MONTHLY: return ("monthly commission");
        case DEAL_TYPE_COMMISSION_AGENT_DAILY: return ("daily agent commission");
        case DEAL_TYPE_COMMISSION_AGENT_MONTHLY: return ("monthly agent commission");
        case DEAL_TYPE_INTEREST:          return ("interest rate");
        case DEAL_TYPE_BUY_CANCELED:      descr="cancelled buy deal"; break;
        case DEAL_TYPE_SELL_CANCELED:     descr="cancelled sell deal"; break;
    }
    descr=StringFormat("%s %G %s (order #%d, position ID %d)",
        descr, // laufender Zustand
        volume, // Dealsvolumen
        symbol, // Symbol des Deals
        ticket, // Ticket der Order, die den Deal aufgerufen hat
        pos_ID // ID der Position des Deals
    );

    return(descr);
//---
}

```

#### Sehen Sie auch

[HistorySelect\(\)](#), [HistoryDealsTotal\(\)](#), [HistoryDealSelect\(\)](#), [Eigenschaften der Deals](#)

## HistoryDealGetDouble

Gibt die angeforderte Eigenschaft des Deals zurück. Eigenschaft des Deals muss des Typs double sein. Es gibt 2 Varianten der Funktion.

1. Gibt den Wert der Eigenschaft sofort zurück.

```
double HistoryDealGetDouble(  
    ulong          ticket_number,      // Ticket  
    ENUM_DEAL_PROPERTY_DOUBLE property_id // Identifikator der Eigenschaft  
);
```

2. Gibt true oder false zurück abhängig davon, ob die Funktion erfolgreich durchgeführt wird. Im Erfolgsfall wird der Wert der Eigenschaft in die Empfangsvariable gesetzt, die durch Referenz vom letzten Parameter übertragen wird.

```
bool HistoryDealGetDouble(  
    ulong          ticket_number,      // Ticket  
    ENUM_DEAL_PROPERTY_DOUBLE property_id, // Identifikator der Eigenschaft  
    double&        double_var         // hier nehmen wir den Wert der Eigenschaft  
);
```

### Parameter

*ticket\_number*

[in] Ticket des Deals.

*property\_id*

[in] Identifikator der Eigenschaft des Deals. Wert kann einer der Aufzählungswerte [ENUM\\_DEAL\\_PROPERTY\\_DOUBLE](#) sein.

*double\_var*

[out] Variable des Typs double, die den Wert der angeforderten Eigenschaft annimmt.

### Rückgabewert

Wert des Typs [double](#).

### Bemerkung

Man muss nicht [Ordern](#), [Deals](#) und [Positionen](#) verwechseln. Jeder Deal ist das Ergebnis der Ausführung einer Order, jede Position ist das Abschlussergebnis eines oder mehrerer Deals.

### Sehen Sie auch

[HistoryDealsTotal\(\)](#), [HistorySelect\(\)](#), [HistoryDealGetTicket\(\)](#), [Eigenschaften der Deals](#)

## HistoryDealGetInteger

Gibt die angeforderte Eigenschaft des Deals zurück. Eigenschaft des Deals muss des Typs `datetime`, `int` sein. Es gibt 2 Varianten der Funktion.

1. Gibt den wert der Eigenschaft sofort zurück.

```
long HistoryDealGetInteger(  
    ulong          ticket_number,    // Ticket  
    ENUM_DEAL_PROPERTY_INTEGER property_id // Identifikator der Eigenschaft  
);
```

2. Gibt `true` oder `false` zurück abhängig davon, ob die Funktion erfolgreich durchgeführt wird. Im Erfolgsfall wird der Wert der Eigenschaft in die Empfangsvariable gesetzt, die durch Referenz vom letzten Parameter übertragen wird.

```
bool HistoryDealGetInteger(  
    ulong          ticket_number,    // Ticket  
    ENUM_DEAL_PROPERTY_INTEGER property_id, // Identifikator der Eigenschaft  
    long&          long_var          // hier nehmen wir den Wert der Eigenschaft  
);
```

### Parameter

*ticket\_number*

[in] Ticket des Deals.

*property\_id*

[in] Identifikator der Eigenschaft des Deals. Wert kann einer der Aufzählungswerte [ENUM\\_DEAL\\_PROPERTY\\_INTEGER](#) sein.

*long\_var*

[out] Wert des Typs [long](#).

### Rückgabewert

Wert des Typs [long](#).

### Bemerkung

Man muss nicht [Ordern](#), [Deals](#) und [Positionen](#) verwechseln. Jeder Deal ist das Ergebnis der Ausführung einer Order, jede Position ist das Abschlussergebnis eines oder mehrerer Deals.

### Beispiel:

```

//+-----+
//| Trade function |
//+-----+
void OnTrade()
{
//--- Ticket des letzten Deals aus der Handelsgeschichte für eine Woche erhalten
ulong last_deal=GetLastDealTicket();
if(HistoryDealSelect(last_deal))
{
//--- Zeit des Deals in Millisekunden seit 01.01.1970
long deal_time_msc=HistoryDealGetInteger(last_deal,DEAL_TIME_MSC);
PrintFormat("Deal #d DEAL_TIME_MSC=%i64 => %s",
            last_deal,deal_time_msc,TimeToString(deal_time_msc/1000));
}
else
PrintFormat("HistoryDealSelect() failed for #d. Error code=%d",
            last_deal,GetLastError());
//---
}
//+-----+
//| Gibt Ticket des letzten Deals in der Geschichte oder -1 zurück |
//+-----+
ulong GetLastDealTicket()
{
//--- Geschichte für die letzte 7 Tagen erfordern
if(!GetTradeHistory(7))
{
//--- Nachricht über fehlgeschlagenen Aufruf und Rückgabe -1
Print(__FUNCTION__," HistorySelect() hat false zurückgegeben");
return -1;
}
//---
ulong first_deal,last_deal,deals=HistoryOrdersTotal();
//--- wenn es Ordern gibt, anfangen wir mit ihnen zu arbeiten
if(deals>0)
{
Print("Deals = ",deals);
first_deal=HistoryDealGetTicket(0);
PrintFormat("first_deal = %d",first_deal);
if(deals>1)
{
last_deal=HistoryDealGetTicket((int)deals-1);
PrintFormat("last_deal = %d",last_deal);
return last_deal;
}
return first_deal;
}
//--- keine Deals waren gefunden, Rückgabe -1
return -1;
}
//+-----+
//| Geschichte für letzte Tage erfordert und gibt false beim Fehler zurück |
//+-----+
bool GetTradeHistory(int days)
{
//--- definieren wir Wochenperiode, um eine Geschichte des Handels zu beantragen
datetime to=TimeCurrent();
datetime from=to-days*PeriodSeconds(PERIOD_D1);
ResetLastError();
//--- Machen wir eine Anfrage und überprüfen wir die Ergebnisse
if(!HistorySelect(from,to))

```

```
{
    Print(__FUNCTION__," HistorySelect=false. Error code=",GetLastError());
    return false;
}
//--- Geschichte erfolgreich empfangen
return true;
}
```

**Sehen Sie auch**

[HistoryDealsTotal\(\)](#), [HistorySelect\(\)](#), [HistoryDealGetTicket\(\)](#), [Eigenschaften der Deals](#)



## HistoryDealGetString

Gibt die angeforderte Eigenschaft des Deals zurück. Eigenschaft des Deals muss des des Typs string sein. Es gibt 2 Varianten der Funktion.

1. Gibt den Wert der Eigenschaft sofort zurück.

```
string HistoryDealGetString(
    ulong          ticket_number,    // Ticket
    ENUM_DEAL_PROPERTY_STRING property_id // Identifikator der Eigenschaft
);
```

2. Gibt true oder false zurück abhängig davon, ob die Funktion erfolgreich durchgeführt wird. Im Erfolgsfall wird der Wert der Variable in die Empfangsvariable gesetzt, die durch Referenz vom letzten Parameter übertagen wird.

```
bool HistoryDealGetString(
    ulong          ticket_number,    // Ticket
    ENUM_DEAL_PROPERTY_STRING property_id, // Identifikator der Eigenschaft
    string&        string_var       // hier nehmen wir den Wert der Eigenschaft
);
```

### Parameter

*ticket\_number*

[in] Ticket des Deals.

*property\_id*

[in] Identifikator der Eigenschaft des Deals. Wert kann einer der Aufzählungswerte [ENUM\\_DEAL\\_PROPERTY\\_STRING](#) sein.

*string\_var*

[out] Variable des Typs string, die den Wert der angeforderten Eigenschaft annimmt.

### Rückgabewert

Wert des Typs [string](#).

### Bemerkung

Man muss nicht [Ordern](#), [Deals](#) und [Positionen](#) verwechseln. Jeder Deal ist das Ergebnis der Ausführung einer Order, jede Position ist das Abschlussergebnis eines oder mehrerer Deals.

### Sehen Sie auch

[HistoryDealsTotal\(\)](#), [HistorySelect\(\)](#), [HistoryDealGetTicket\(\)](#), [Eigenschaften der Deals](#)

## Handelssignale

Eine Gruppe von Funktionen für die Steuerung der Handelssignale. Diese Funktionen erlauben Ihnen:

- Informationen über die Handelssignale, die zum Kopieren zur Verfügung stehen, erhalten,
- Kopiereinstellungen von Handelssignale lesen oder setzen,
- zu einem Signal abonnieren und Subskription durch die Sprache MQL5 abschaffen.

| Funktion                                    | Aktion                                                                                               |
|---------------------------------------------|------------------------------------------------------------------------------------------------------|
| <a href="#"><u>SignalBaseGetDouble</u></a>  | Gibt den Wert des Eigenschafts vom Typ double für das gewählte Signal zurück                         |
| <a href="#"><u>SignalBaseGetInteger</u></a> | Gibt den Wert des Eigenschafts vom Typ integer für das gewählte Signal zurück                        |
| <a href="#"><u>SignalBaseGetString</u></a>  | Gibt den Wert des Eigenschafts vom Typ string für das gewählte Signal zurück                         |
| <a href="#"><u>SignalBaseSelect</u></a>     | Wählt ein Signal aus der Basis der Handelssignale, die im Terminal erhältlich sind                   |
| <a href="#"><u>SignalBaseTotal</u></a>      | Gibt die Anzahl der im Terminal verfügbare Signale zurück                                            |
| <a href="#"><u>SignalInfoGetDouble</u></a>  | Gibt den Wert des Eigenschafts vom Typ double aus der Kopiereinstellungen des Handelssignals zurück  |
| <a href="#"><u>SignalInfoGetInteger</u></a> | Gibt den Wert des Eigenschafts vom Typ integer aus der Kopiereinstellungen des Handelssignals zurück |
| <a href="#"><u>SignalInfoGetString</u></a>  | Gibt den Wert des Eigenschafts vom Typ string aus der Kopiereinstellungen des Handelssignals zurück  |
| <a href="#"><u>SignalInfoSetDouble</u></a>  | Setzt den Wert des Eigenschafts vom Typ double in Kopiereinstellungen des Handelssignals             |
| <a href="#"><u>SignalInfoSetInteger</u></a> | Setzt den Wert des Eigenschafts vom Typ integer in Kopiereinstellungen des Handelssignals            |
| <a href="#"><u>SignalSubscribe</u></a>      | Abonniert auf ein Handelssignal                                                                      |
| <a href="#"><u>SignalUnsubscribe</u></a>    | Abbestellt Subscription auf ein Handelssignal                                                        |

## SignalBaseGetDouble

Возвращает значение свойства типа [double](#) для выбранного сигнала.

```
double SignalBaseGetDouble (
    ENUM_SIGNAL_BASE_DOUBLE    property_id,    // идентификатор свойства
);
```

### Параметры

*property\_id*

[in] Идентификатор свойства сигнала. Может быть одним из значений перечисления [ENUM\\_SIGNAL\\_BASE\\_DOUBLE](#).

### Возвращаемое значение

Значение типа [double](#) указанного свойства сигнала.

## SignalBaseGetInteger

Возвращает значение свойства типа [integer](#) для выбранного сигнала.

```
long SignalBaseGetInteger (
    ENUM_SIGNAL_BASE_INTEGER    property_id,    // идентификатор свойства
);
```

### Параметры

*property\_id*

[in] Идентификатор свойства сигнала. Может быть одним из значений перечисления [ENUM\\_SIGNAL\\_BASE\\_INTEGER](#).

### Возвращаемое значение

Значение типа [integer](#) указанного свойства сигнала.

## SignalBaseGetString

Возвращает значение свойства типа [string](#) для выбранного сигнала.

```
string SignalBaseGetString(  
    ENUM_SIGNAL_BASE_STRING    property_id,    // идентификатор свойства  
);
```

### Параметры

*property\_id*

[in] Идентификатор свойства сигнала. Может быть одним из значений перечисления [ENUM\\_SIGNAL\\_BASE\\_STRING](#).

### Возвращаемое значение

Значение типа [string](#) указанного свойства сигнала.

## SignalBaseSelect

Выбирает для работы сигнал из базы торговых сигналов, доступных в терминале.

```
bool SignalBaseSelect(  
    int     index    // индекс записи сигнала  
);
```

### Параметры

*index*

[in] Индекс записи сигнала в базе торговых сигналов.

### Возвращаемое значение

Возвращает true при успешном завершении функции или false в случае ошибки. Чтобы получить информацию об [ошибке](#), необходимо вызвать функцию [GetLastError\(\)](#).

### Пример:

```
void OnStart()  
{  
    //--- запрашиваем общее количество сигналов в базе  
    int total=SignalBaseTotal();  
    //--- цикл по всем сигналам  
    for(int i=0;i<total;i++)  
    {  
        //--- выбираем сигнал для дальнейшей работы  
        if(SignalBaseSelect(i))  
        {  
            //--- получение свойств сигнала  
            long id    =SignalBaseGetInteger(SIGNAL_BASE_ID);           // id сигнала  
            long pips   =SignalBaseGetInteger(SIGNAL_BASE_PIPS);        // результат торг  
            long subscr=SignalBaseGetInteger(SIGNAL_BASE_SUBSCRIBERS);  // количество по  
            string name =SignalBaseGetString(SIGNAL_BASE_NAME);        // имя сигнала  
            double price=SignalBaseGetDouble(SIGNAL_BASE_PRICE);       // цена подписки  
            string curr =SignalBaseGetString(SIGNAL_BASE_CURRENCY);    // валюта сигнал  
            //--- выводим все прибыльные бесплатные сигналы с ненулевым количеством подп  
            if(price==0.0 && pips>0 && subscr>0)  
                PrintFormat("id=%d, name=\"%s\", currency=%s, pips=%d, subscribers=%d",id,  
                            );  
        }  
        else PrintFormat("Ошибка выбора сигнала. Код ошибки=%d",GetLastError());  
    }  
}
```

## SignalBaseTotal

Возвращает общее количество сигналов, доступных в терминале.

```
int SignalBaseTotal();
```

### Возвращаемое значение

Общее количество сигналов, доступных в терминале.

## SignalInfoGetDouble

Возвращает из настроек копирования торгового сигнала значение свойства типа [double](#).

```
double SignalInfoGetDouble(  
    ENUM_SIGNAL_INFO_DOUBLE    property_id,    // идентификатор свойства  
);
```

### Параметры

*property\_id*

[in] Идентификатор свойства настроек копирования торгового сигнала. Может быть одним из значений перечисления [ENUM\\_SIGNAL\\_INFO\\_DOUBLE](#).

### Возвращаемое значение

Значение типа [double](#) указанного свойства настроек копирования торгового сигнала.



## SignalInfoGetInteger

Возвращает из настроек копирования торгового сигнала значение свойства типа [integer](#).

```
long  SignalInfoGetInteger (
    ENUM_SIGNAL_INFO_INTEGER  property_id,    // идентификатор свойства
);
```

### Параметры

*property\_id*

[in] Идентификатор свойства настроек копирования торгового сигнала. Может быть одним из значений перечисления [ENUM\\_SIGNAL\\_INFO\\_INTEGER](#).

### Возвращаемое значение

Значение типа [integer](#) указанного свойства настроек копирования торгового сигнала.

## SignalInfoGetString

Возвращает из настроек копирования торгового сигнала значение свойства типа [string](#).

```
string SignalInfoGetString(  
    ENUM_SIGNAL_INFO_STRING    property_id,    // идентификатор свойства  
);
```

### Параметры

*property\_id*

[in] Идентификатор свойства настроек копирования торгового сигнала. Значение может быть одним из значений перечисления [ENUM\\_SIGNAL\\_INFO\\_STRING](#).

### Возвращаемое значение

Значение типа [string](#) указанного свойства настроек копирования торгового сигнала.

## SignalInfoSetDouble

Устанавливает в настройках копирования торгового сигнала значение свойства типа [double](#).

```
bool SignalInfoSetDouble (
    ENUM_SIGNAL_INFO_DOUBLE    property_id,    // идентификатор свойства
    double                     value           // значение свойства
);
```

### Параметры

*property\_id*

[in] Идентификатор свойства настроек копирования торгового сигнала. Может быть одним из значений перечисления [ENUM\\_SIGNAL\\_INFO\\_DOUBLE](#).

*value*

[in] Значение свойства настроек копирования торгового сигнала.

### Возвращаемое значение

Возвращает true в случае успешного изменения свойства, иначе возвращает false. Чтобы получить дополнительную информацию об [ошибке](#), необходимо вызвать функцию [GetLastError\(\)](#).

## SignalInfoSetInteger

Устанавливает в настройках копирования торгового сигнала значение свойства типа [integer](#).

```
bool SignalInfoSetInteger (
    ENUM_SIGNAL_INFO_INTEGER    property_id,    // идентификатор свойства
    long                        value           // значение свойства
);
```

### Параметры

*property\_id*

[in] Идентификатор свойства настроек копирования торгового сигнала. Может быть одним из значений перечисления [ENUM\\_SIGNAL\\_INFO\\_INTEGER](#).

*value*

[in] Значение свойства настроек копирования торгового сигнала.

### Возвращаемое значение

Возвращает true в случае успешного изменения свойства, иначе возвращает false. Чтобы получить дополнительную информацию об [ошибке](#), необходимо вызвать функцию [GetLastError\(\)](#).

## SignalSubscribe

Производит подписку на копирование указанного торгового сигнала.

```
bool SignalSubscribe(  
    long    signal_id    // id сигнала  
);
```

### Параметры

*signal\_id*

[in] Идентификатор сигнала.

### Возвращаемое значение

Возвращает true в случае успешной подписки на копирование торгового сигнала, иначе возвращает false. Чтобы получить дополнительную информацию об [ошибке](#), необходимо вызвать функцию [GetLastError\(\)](#).

## SignalUnsubscribe

Отменяет подписку на копирование торгового сигнала.

```
bool SignalUnsubscribe();
```

### Возвращаемое значение

Возвращает true в случае успешной отмены подписки на копирование торгового сигнала, иначе возвращает false. Чтобы получить дополнительную информацию об [ошибке](#), необходимо вызвать функцию [GetLastError\(\)](#).

## Globalvariablen des Kundenterminals

Gruppe der Funktionen, die für Arbeit mit Globalvariablen bestimmt sind.

Globale Variablen des Kundenterminals müssen nicht mit den im [globalen\\_Massstab](#) erklärten Variablen des mql5-Programms verwechselt werden.

Globale Variablen befinden sich im Kundenterminal innerhalb von 4 Wochen seit dem letzten Zugang, dann werden sie automatisch entfernt. Zugang zur globalen Variable ist nicht nur die Einstellung des neuen Wertes, sondern auch Lesen des Wertes der globalen Variable.

Globale Variablen des Kundenterminals sind gleichzeitig aus allen mql5-Programmen zugänglich, die im Kundenterminal gestartet sind.

| Funktion                                     | Massnahme                                                                                                 |
|----------------------------------------------|-----------------------------------------------------------------------------------------------------------|
| <a href="#">GlobalVariableCheck</a>          | Prüft, ob die globale Variable mit dem angegebenen Namen existiert                                        |
| <a href="#">GlobalVariableTime</a>           | Gibt die Zeit des letzten Zuganges zur globalen Variable zurück                                           |
| <a href="#">GlobalVariableDel</a>            | entfernt globale Variable                                                                                 |
| <a href="#">GlobalVariableGet</a>            | Fordert den Wert der globalen Variable an                                                                 |
| <a href="#">GlobalVariableName</a>           | Gibt den Namen der globalen Variable nach der laufenden Nummer in der Liste der globalen Variable         |
| <a href="#">GlobalVariableSet</a>            | Stellt den neuen Wert der globalen Variable fest                                                          |
| <a href="#">GlobalVariablesFlush</a>         | Speichert den Inhalt aller Variablen auf der Platte zwangsmaessig                                         |
| <a href="#">GlobalVariableTemp</a>           | Stellt den neuen wert der globalen Variable fest, die nur während der laufenden Terminalsitzung existiert |
| <a href="#">GlobalVariableSetOnCondition</a> | Stellt den neuen Wert der existierenden globalen Variable nach Bedingung                                  |
| <a href="#">GlobalVariablesDeleteAll</a>     | Entfernt globale Variablen mit dem angegebenen Praefix im Namen                                           |
| <a href="#">GlobalVariablesTotal</a>         | Gibt die ganze Zahl der globalen Variablen zurück                                                         |

Glo

## GlobalVariableCheck

Prueft das Existieren der globalen Variable des Kundenterminals.

```
bool GlobalVariableCheck(  
    string name    // Name  
);
```

### Parameter

*name*

[in] Name der globalen Variable.

### Rückgabewert

Gibt de Wert true zurück, wenn die globale Variable existiert, anderenfalls gibt false zurück.

### Bemerkung

Globale Variablen existieren im Kundenterminal innerhalb von 4 Wochen seit dem letzten Zugriffs, danach werden sie automatisch entfernt.

### Sehen Sie auch

[GlobalVariableTime\(\)](#)



## GlobalVariableTime

Gibt die Zeit des letzten Zugangs zur globalen Variable zurück.

```
datetime GlobalVariableTime(  
    string name    // Name  
);
```

### Parameter

*name*

[in] Name der globalen Variable.

### Rückgabewert

Gibt die Zeit des letzten Zugangs zu angegebenen globalen Variable. Aufruf der angegebenen globalen Variable ist auch Zugang. Für die Erhaltung der [fehlerbezogenen](#) Information muss die Funktion [GetLastError\(\)](#) aufgerufen werden.

### Bemerkung

Globale Variablen existieren im Kundenterminal innerhalb von 4 Wochen seit dem letzten Zugriffs, danach werden sie automatisch entfernt.

### Sehen Sie auch

[GlobalVariableCheck\(\)](#)

## GlobalVariableDel

Entfernt globale Variable des Kundenterminals.

```
bool GlobalVariableDel(  
    string name    // Name  
);
```

### Parameter

*name*

[in] Name der globalen Variable.

### Rückgabewert

Bei der erfolgreichen Entfernung gibt die Funktion true zurück, anderenfalls false. Für die Erhaltung der [fehlerbezogenen](#) Information muss die Funktion [GetLastError\(\)](#) aufgerufen werden.

### Bemerkung

Globale Variablen existieren im Kundenterminal innerhalb von 4 Wochen seit dem letzten Zugriffs, danach werden sie automatisch entfernt.

## GlobalVariableGet

Gibt den Wert der globalen Variable des Kundenterminals zurück. Es gibt zwei Varianten der Funktion.

1. Gibt den Wert der Eigenschaft sofort zurück.

```
double GlobalVariableGet(  
    string name      // Name  
);
```

2. Gibt true oder false zurück abhängig davon, ob die Funktion erfolgreich durchgeführt wurde. Im Erfolgsfall wird der Wert der Variable des Kundenterminals in die Empfangsvariable gesetzt, die durch Referenz vom letzten Parameter übertragen wird.

```
bool GlobalVariableGet(  
    string name          // Name  
    double& double_var   // hier nehmen wir den Wert der globalen Variable auf  
);
```

### Parameter

*name*

[in] Name der globalen Variable.

*double\_var*

[out] Variable des Typs double, die den Wert annimmt, der in der globalen Variable des Kundenterminals aufbewahrt wird.

### Rückgabewert

Wert der existierenden globalen Variable oder 0 beim [Fehler](#). Für die Erhaltung der fehlerbezogenen Information muss die Funktion [GetLastError\(\)](#) aufgerufen werden.

### Bemerkung

Globale Variablen existieren im Kundenterminal innerhalb von 4 Wochen seit dem letzten Zugriffs, danach werden sie automatisch entfernt.

## GlobalVariableName

Gibt den Wert der globalen Variable nach der Ordnungsnummer zurück.

```
string GlobalVariableName (  
    int index      // Nummer in der Liste der globalen Variablen  
);
```

### Parameter

*index*

[in] Ordnungsnummer in der Liste der globalen Variablen. Muss grösser als oder 0 sein und weniger als [GlobalVariablesTotal\(\)](#).

### Rückgabewert

Name der globalen Variable nach der Ordnungsnummer in der Liste der globalen Variablen. Für die Erhaltung der [fehlerbezogenen](#) Information muss die Funktion [GetLastError\(\)](#) aufgerufen werden.

### Bemerkung

Globale Variablen existieren im Kundenterminal innerhalb von 4 Wochen seit dem letzten Zugriff, danach werden sie automatisch entfernt.

## GlobalVariableSet

Stellt einen neuen Wert der globalen Variable. Wenn die Variable existiert nicht, erzeugt das System eine neue Variable.

```
datetime GlobalVariableSet(  
    string  name,           // Name  
    double  value           // der festgestellte Wert  
);
```

### Parameter

*name*

[in] Name der globalen Variable.

*value*

[in] Neuer numerische Wert.

### Rückgabewert

Bei der erfolgreichen Durchführung gibt die Funktion die Zeit des letzten Zugangs, anderenfalls 0. Für die Erhaltung der [fehlerbezogenen](#) Information muss die Funktion [GetLastError\(\)](#) aufgerufen werden.

### Bemerkung

Globale Variablen existieren im Kundenterminal innerhalb von 4 Wochen seit dem letzten Zugriff, danach werden sie automatisch entfernt.

## GlobalVariablesFlush

Speichert den Inhalt aller globalen Variablen in der Platte zwangsmaessig.

```
void GlobalVariablesFlush();
```

### Rückgabewert

Keinen Rückgabewert.

### Bemerkung

Das Terminal speichert alle globalen Variablen, wenn die Arbeit zu Ende ist, aber Daten können verloren werden beim ploetzlichen Fehler des Computers. Diese Funktion erlaubt das Prozess der Speicherung der globalen Variablen selbststaendig zu verwalten, falls unvorhergesehene Ereignisse vorkommen.

## GlobalVariableTemp

Versucht, temporäre Variable zu erzeugen. Wenn die Variable existiert nicht, erzeugt das System eine neue temporäre Variable.

```
bool GlobalVariableTemp(  
    string name,      // Name  
);
```

### Parameter

*name*

[in] Name der temporären globalen Variable.

### Rückgabewert

Bei der erfolgreichen Durchführung gibt die Funktion true zurück, anderenfalls false. Für die Erhaltung der [fehlerbezogenen](#) Information, muss die Funktion [GetLastError\(\)](#) aufgerufen werden.

### Bemerkung

Temporäre globale Variablen existieren nur während der Session des Kundenterminals, nach Schliessen des Terminals werden sie automatisch entfernt. Bei der Durchführung der Operation [GlobalVariablesFlush\(\)](#) werden globale Variablen in der Platte nicht gespeichert.

Nach Erzeugung der temporären globalen Variable wird der Zugang zu ihr und ihre Modifikation ebenso, wie zur normalen [globalen Variable des Kundenterminals](#) durchgeführt..

## GlobalVariableSetOnCondition

Stellt einen neuen Wert der existierenden globalen Variable, wenn der laufende Wert der Variable dem Wert des dritten Parameters `check_value` gleich wird. Wenn es keine globale Variable gibt, wird die Funktion den Fehler `ERR_GLOBALVARIABLE_NOT_FOUND` (4501) generieren und `false` zurückgeben.

```
bool GlobalVariableSetOnCondition(  
    string name,           // Name  
    double value,          // Wert bei der Erfuellung der Bedingung  
    double check_value     // die geprüfte Bedingung  
);
```

### Parameter

*name*

[in] Name der globalen Variable.

*value*

[in] Neuer Wert.

*check\_value*

[in] Wert für die Prüfung des laufenden Wertes der globalen Variable.

### Rückgabewert

Die Funktion gibt `true` zurück, anderenfalls `false`. Für die Erhaltung der [fehlerbezogenen](#) Information muss die Funktion [GetLastError\(\)](#) aufgerufen werden. Wenn sich der laufende Wert der globalen Variable von `check_value` unterscheidet, gibt die Funktion `false` zurück.

### Bemerkung

Funktion gewährt automaten Zugang zur globalen Variable, darum kann sie Semaphore gewähren bei der Interaktion der Experten, die gleichzeitig innerhalb eines Kundenterminals arbeiten.



## GlobalVariablesDeleteAll

Entfernt globale Variablen des Kundenterminals.

```
int GlobalVariablesDeleteAll(  
    string    prefix_name=NULL    // alle globalen Variablen, deren Namen mit Praefi  
    datetime  limit_data=0        // alle globalen Variablen, die vor diesem Datum v  
);
```

### Parameter

*prefix\_name=NULL*

[in] Praefixname der entfernten globalen Variablen. Wenn das Praefix NULL oder Leerzeile angegeben wird, werden alle globalen Variablen, die dem Datumkriterium entsprechen, entfernt.

*limit\_data=0*

[in] Datum für Auswahl der globalen Variablen nach der Zeit der letzten Modifikation. Alle Variablen, die vor dem angegebenen Datum verändert wurden, werden entfernt. Wenn der Parameter gleich Null ist, werden alle Variablen, die dem ersten Kriterium entsprechen (Praefix), entfernt.

### Rückgabewert

Anzahl der entfernten Variablen.

### Bemerkung

Wenn die beiden Parameter gleich Null sind (*prefix\_name=NULL* und *limit\_data=0*), werden alle globale Variablen des Kundenterminals entfernt. Wenn beide Parameter angegeben werden, werden globale Variablen, die beiden Parametern entsprechen, entfernt.

Globale Variablen existieren im Kundenterminal innerhalb von 4 Wochen seit dem letzten Zugriffs, danach werden sie automatisch entfernt.

## GlobalVariablesTotal

Gibt die gesamte Anzahl der globalen Variablen des Kundenterminals zurück.

```
int GlobalVariablesTotal();
```

### Rückgabewert

Anzahl der globalen Variablen.

### Bemerkung

Globale Variablen existieren im Kundenterminal innerhab von 4 Wochen seit dem letzten Zugriffs, danach werden sie automatisch entfernt. Zugriff zur globalen Variable ist nicht nur die Einstellung des neuen Wertes, sondern auch das Lesen des Wertes der globalen Variable.

## Dateioperationen

Gruppe der Funktionen für Arbeit mit Dateien.

Aus Sicherheitsgründen ist die Arbeit mit Dateien in MQL5 streng gesteuert. Dateien, mit denen Datei-Operationen mittels MQL5 durchgeführt werden, können nicht außerhalb der Datei-Sandbox sein.

Es gibt zwei Verzeichnisse (mit Subverzeichnissen), in denen Arbeitsdateien befinden können:

- Terminal\_data\_folder\MQL5\FILES\ (Im Menue des Terminals wählen Sie für Durchsicht Manüpunkte "Datei"->"Datenverzeichnis öffnen");
- der gesamter Ordner aller im Computer installierten Terminals - befinden sich normalerweise im Verzeichnis C:\Documents and Settings\All Users\Application Data\MetaQuotes\Terminal\Common\Files.

Durch das Programmverzeichnis kann man Benennungen dieser dieser Verzeichnisse durch die Funktion [TerminalInfoString\(\)](#) erhalten werden, beim Verwenden von Aufzählungen [ENUM\\_TERMINAL\\_INFO\\_STRING](#):

```
//--- Ordner, in dem Terminaldaten aufbewahren werden
string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
//--- der Gesamtordner aller Kundenterminals
string common_data_path=TerminalInfoString(TERMINAL_COMMONDATA_PATH);
```

Arbeit mit Dateien aus anderen Verzeichnissen wird untersagt.

Die Datei Funktionen können Sie mit sogenannten "die genannten Kanäle" arbeiten. Dazu ist genügend es, die Funktion [FileOpen\(\)](#) mit den entsprechenden Parametern zu rufen</t2.

| Funktion                         | Massnahme                                                                                |
|----------------------------------|------------------------------------------------------------------------------------------|
| <a href="#">FileFindFirst</a>    | Fängt die Suche der Dateien im entsprechenden Verzeichnis nach dem angegebenen Filter an |
| <a href="#">FileFindNext</a>     | Setzt die Suche fort, die durch die Funktion FileFindFirst() angefangen wurde            |
| <a href="#">FileFindClose</a>    | Schließt die Suche Handle                                                                |
| <a href="#">FileOpen</a>         | Öffnet die Datei mit dem angegebenen Namen und mit der angegebenen Flagge                |
| <a href="#">FileDelete</a>       | Entfernt die angegebene Datei                                                            |
| <a href="#">FileFlush</a>        | Speicher auf der Platte alle Daten, die im Dateipuffer Eingabe-Ausgabe noch bleiben      |
| <a href="#">FileGetInteger</a>   | Ruft eine ganzzahlige Eigenschaft der Datei                                              |
| <a href="#">FileIsEnding</a>     | Bestimmt das Ende der Datei im während des Lesens                                        |
| <a href="#">FileIsLineEnding</a> | Bestimmt Zeilenende während des Lesens                                                   |
| <a href="#">FileClose</a>        | Schließt die früher geöffnete Datei                                                      |

|                                         |                                                                                                                                                                                                |
|-----------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#"><u>FileIsExist</u></a>      | Prüft, ob die Datei existiert                                                                                                                                                                  |
| <a href="#"><u>FileCopy</u></a>         | Kopiert die Ausgangsdatei aus dem lokalen oder gesamten Ordner in eine andere Datei                                                                                                            |
| <a href="#"><u>FileMove</u></a>         | Bewegt oder benennt die Datei um                                                                                                                                                               |
| <a href="#"><u>FileReadArray</u></a>    | Liest Arrays verschiedener Typen außer Zeilentypen (es kann ein Strukturarray sein ohne Zeilen und die dynamische Arrays) aus der binären Datei von der laufenden Position des Dateiindikators |
| <a href="#"><u>FileReadBool</u></a>     | Liest aus der Datei des Typs CSV die Zeile von der laufenden Position bis zum Begrenzer (oder bis zum Ende der Textzeile) und wandelt die gelesene Zeile in den Wert des Typs bool             |
| <a href="#"><u>FileReadDatetime</u></a> | Liest aus der Datei des Typs CSV die Zeile eines der Formate: "YYYY.MM.DD HH:MI:SS", "YYYY.MM.DD" oder "HH:MI:SS" - und wandelt sie in den Wert des Typs datetime um                           |
| <a href="#"><u>FileReadDouble</u></a>   | Liest die Zahl der doppelten Genauigkeit mit dem Fließpunkt (double) aus der binären Datei von der laufenden Position des Dateiindikators                                                      |
| <a href="#"><u>FileReadFloat</u></a>    | Liest den Wert float von der laufenden Position des Dateianzeigers                                                                                                                             |
| <a href="#"><u>FileReadInteger</u></a>  | Liest den Wert des Typs int, short oder char von der laufenden Position des Dateianzeigers abhängig von der angegebenen Länge in Bytes                                                         |
| <a href="#"><u>FileReadLong</u></a>     | Liest den Wert des Typs long von der laufenden Position des Dateianzeigers                                                                                                                     |
| <a href="#"><u>FileReadNumber</u></a>   | Liest aus der Datei des Typs CSV die Zeile von der laufenden Position zum Begrenzer (oder bis zum Ende der Textzeile) und wandelt die gelesene Zeile in den Wert des Typs double um            |
| <a href="#"><u>FileReadString</u></a>   | Liest die Zeile aus der Datei von der laufenden Position des Dateianzeigers                                                                                                                    |
| <a href="#"><u>FileReadStruct</u></a>   | Liest Inhalt aus der binären Datei in die Struktur, die als Parameter übertragen wurde                                                                                                         |
| <a href="#"><u>FileSeek</u></a>         | Bewegt die Position des Dateiindikators um die angegebene Zahl der Bytes gegen die angegebene Position                                                                                         |
| <a href="#"><u>FileSize</u></a>         | Gibt die Größe der entsprechenden geöffneten Datei zurück                                                                                                                                      |
| <a href="#"><u>FileTell</u></a>         | Gibt die laufende Position des Indikators der entsprechenden geöffneten Datei zurück                                                                                                           |

|                                         |                                                                                                                                         |
|-----------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#"><u>FileWrite</u></a>        | Schreibt Daten in die Datei des Typs CSV oder TXT                                                                                       |
| <a href="#"><u>FileWriteArray</u></a>   | Schreibt in die Datei des Typs BIN Arrays aller Typen außer Zeilentypen                                                                 |
| <a href="#"><u>FileWriteDouble</u></a>  | Schreibt den Parameterwert des Typs double von der angegebenen Position des Dateianzeigers in die binäre Datei                          |
| <a href="#"><u>FileWriteFloat</u></a>   | Schreibt den Parameterwert des Typs float von der angegebenen Position des Dateianzeigers                                               |
| <a href="#"><u>FileWriteInteger</u></a> | Schreibt den Parameterwert des Typs int in die binäre Datei von der laufenden Position des Dateianzeigers                               |
| <a href="#"><u>FileWriteLong</u></a>    | Schreibt den Parameterwert des Typs long in die binäre Datei von der laufenden Position des Dateianzeigers                              |
| <a href="#"><u>FileWriteString</u></a>  | Schreibt den Parameterwert des Typs string in die Datei des Typs BIN oder TXT von der laufenden Position des Dateianzeigers             |
| <a href="#"><u>FileWriteStruct</u></a>  | Schreibt den Inhalt der Struktur, die als Parameter übertragen wurde, in die binäre Datei von der laufenden Position des Dateianzeigers |
| <a href="#"><u>FolderCreate</u></a>     | Erzeugt ein Verzeichnis im Ordner Files (abhängig von dem Wert common_flag)                                                             |
| <a href="#"><u>FolderDelete</u></a>     | Entfernt das angegebene Verzeichnis. Wenn der Ordner nicht leer ist, kann er entfernt werden                                            |
| <a href="#"><u>FolderClean</u></a>      | Entfernt alle Dateien im angegebenen Ordner                                                                                             |

Wenn Datei für Schreiben durch die Funktion [FileOpen\(\)](#) geöffnet wird, werden alle im Pfad angegebenen Unterordner erzeugt werden, wenn sie fehlen.

## FileFindFirst

Starten die Sortierung der Dateien und der Unterverzeichnisse im entsprechenden Verzeichnis mit dem angegebenen Filter.

```
long FileFindFirst(  
    const string    file_filter,           // Zeile - Suchefilter  
    string&         returned_filename,     // der Name von gefundenen Datei oder Unterver  
    int             common_flag=0         // bestimmt den Suchbereich  
);
```

### Parameter

*file\_filter*

[in] Suchfilter. Im Filter kann Unterverzeichnis (oder Folge der verschachtelten Unterverzeichnisse) gegenüber dem Verzeichnis angegeben werden \Files, in dem Dateien gesucht werden müssen.

*returned\_filename*

[out] Der zurückgegebene Parameter, wohin im Falle des Erfolges der Name von gefundenen Datei oder Unterverzeichnis unterbracht wird.

*common\_flag*

[in] [Flagge](#), die Dateilage bestimmt. Wenn `common_flag=FILE_COMMON` ist, dann befindet sich die Datei im gesamten Ordner aller Kundenterminals \Terminal\Common\Files. Anderenfalls befindet sich die Datei im lokalen Ordner.

### Rückgabewert

Gibt Handle des Objektes der Suche zurück, das man für die weitere Sortierung der Dateien und der Unterverzeichnisse der [FileFindNext\(\)](#) Funktion verwendet werden, oder [INVALID\\_HANDLE](#) wenn es keine Dateien und Unterverzeichnissen gibt, die dem Filter entsprechend (insbesondere, das Verzeichnis ist leer). Nach dem Abschluss der Suche muss man Handle mit Hilfe der [FileFindClose\(\)](#) Funktion schließen.

### Bemerkung

Aus Sicherheitsgründen ist die Arbeit mit Dateien in MQL5 streng gesteuert. Dateien, mit denen Datei-Operationen mittels MQL5 durchgeführt werden, können nicht außerhalb der Datei-Sandbox sein.

### Beispiel:

```

//--- Beim Starten des Scripts zeigen Sie den Fenster mit den Eingabeparametern an
#property script_show_inputs
//--- Filter
input string InpFilter="*";
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    string file_name;
    int i=1;
//--- erhalten Sie die Suche Handle von Grund auf den lokaler Ordner
    long search_handle=FileFindFirst(InpFilter,file_name);
//--- prüfen Sie, ob die FileFindFirst() Funktion erfolgreich durchgeführt hat
    if(search_handle!=INVALID_HANDLE)
    {
        //--- im Zyklus prüfen Sie, ob gesendet Strings Dateinamen oder Verzeichnissen s
        do
        {
            ResetLastError();
            //--- Wenn es die Datei, so wird die Funktion true zurückgeben, und wenn das
            FileIsExist(file_name);
            PrintFormat("%d : %s Name = %s",i,GetLastError()==5018 ? "Verzeichnis" : "Datei",file_name);
            i++;
        }
        while(FileFindNext(search_handle,file_name));
        //--- schließen Sie die Suche Handle
        FileFindClose(search_handle);
    }
    else
        Print("Keine Dateien gefunden!");
}

```

Sehen Sie auch

[FileFindNext](#), [FileFindClose](#)

## FileFindNext

Setzt die Suche fort, die durch die [FileFindFirst\(\)](#) Funktion angefangen wurde.

```
bool FileFindNext (
    long      search_handle,          // die Suche Handle
    string&    returned_filename      // der Name von gefundenen Datei oder Unterverzeichnis
);
```

### Parameter

*search\_handle*

[in] Die Suche Handle, die durch die [FileFindFirst\(\)](#) Funktion erhalten wurde.

*returned\_filename*

[out] Name der nächsten gefundenen Datei oder des Verzeichnisses.

### Rückgabewert

Im Erfolgsfall gibt true zurück, anderenfalls false.

### Beispiel:

```
//--- Beim Starten des Scripts zeigen Sie den Fenster mit den Eingabeparametern an
#property script_show_inputs
//--- Filter
input string InpFilter="*";
//+-----+
//| Script program start function |
//+-----+
void OnStart ()
{
    string file_name;
    int i=1;
    //--- erhalten Sie die Suche Handle von Grund auf den lokaler Ordner
    long search_handle=FileFindFirst(InpFilter,file_name);
    //--- prüfen Sie, ob die FileFindFirst() Funktion erfolgreich durchgeführt hat
    if(search_handle!=INVALID_HANDLE)
    {
        //--- im Zyklus prüfen Sie, ob gesendet Strings Dateinamen oder Verzeichnissen s
        do
        {
            ResetLastError();
            //--- Wenn es die Datei, so wird die Funktion true zurückgeben, und wenn das
            FileIsExist(file_name);
            PrintFormat("%d : %s Name = %s",i,GetLastError()==5018 ? "Verzeichnis" : "Datei",file_name);
            i++;
        }
        while(FileFindNext(search_handle,file_name));
        //--- schließen Sie die Suche Handle
        FileFindClose(search_handle);
    }
    else
        Print("Keine Dateien gefunden!");
}
```

### Sehen Sie auch

[FileFindFirst](#), [FileFindClose](#)



## FileFindClose

Schließen Sie die Suche Handle.

```
void FileFindClose(
    long search_handle    // die Suche Handle
);
```

### Parameter

*search\_handle*

[in] Die Suche Handle, die durch die [FileFindFirst\(\)](#) Funktion erhalten wurde.

### Rückgabewert

Keinen Rückgabewert.

### Bemerkung

Funktion muss aufgerufen werden, um Systemmittel zu befreien.

### Beispiel:

```
/*--- Beim Starten des Scripts zeigen Sie den Fenster mit den Eingabeparametern an
#property script_show_inputs
/*--- Filter
input string InpFilter="*";
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    string file_name;
    int i=1;
    /*--- erhalten Sie die Suche Handle von Grund auf den lokaler Ordner
    long search_handle=FileFindFirst(InpFilter,file_name);
    /*--- prüfen Sie, ob die FileFindFirst()Funktion erfolgreich durchgeführt hat
    if(search_handle!=INVALID_HANDLE)
    {
        /*--- im Zyklus prüfen Sie, ob gesendet Strings Dateinamen oder Verzeichnissen s
        do
        {
            ResetLastError();
            /*--- Wenn es die Datei, so wird die Funktion true zurückgeben, und wenn das
            FileIsExist(file_name);
            PrintFormat("%d : %s Name = %s",i,GetLastError()==5018 ? "Verzeichnis" : "Datei",
            i++,file_name);
        }
        while(FileFindNext(search_handle,file_name));
        /*--- schließen Sie die Suche Handle
        FileFindClose(search_handle);
    }
    else
        Print("Keine Dateien gefunden!");
}
```

### Sehen Sie auch

[FileFindFirst](#), [FileFindNext](#)

## FileIsExist

Prueft, ob die Datei existiert.

```
bool FileIsExist(  
    const string file_name,      // der Dateiname  
    int common_flag=0           // der Suchbereich  
);
```

### Parameter

*file\_name*

[in] Name der geprüften Datei.

*common\_flag=0*

[in] [Flagge](#), die Dateilage bestimmt. Wenn common\_flag=FILE\_COMMON ist, befindet sich die Datei im gesamten Ordner aller Kundenterminals \Terminal\Common\Files. Anderenfalls befindet sich die Datei im lokalen Ordner.

### Rückgabewert

Gibt true zurück, wenn die angegebene Datei existiert.

### Bemerkung

Die prüfen Datei kann ein Unterverzeichnis werden. in diesem Fall gibt die FileIsExist() Funktion false, und in die \_LastError Variable wird Fehler 5018 geschrieben - "Das ist keine Datei, nur Verzeichnis" (sehen Sie das Beispiel für die [FileFindFirst](#) Funktion).

Aus Sicherheitsgründen ist die Arbeit mit Dateien in MQL5 streng gesteuert. Dateien, mit denen Datei-Operationen mittels MQL5 durchgeführt werden, können nicht außerhalb der Datei-Sandbox sein.

Wenn common\_flag=FILE\_COMMON ist, sucht die Funktion die angegebene Datei im Gesamtordner aller Kundenterminals \Terminal\Common\Files, anderenfalls sucht die Funktion die Datei im lokalen Ordner (MQL5\Files oder MQL5\Tester\Files beim Testen).

### Beispiel:

```

//--- Beim Starten des Scripts zeigen Sie den Fenster mit den Eingabeparametern an
#property script_show_inputs
//---Datum für alte Dateien
input datetime InpFilesDate=D'2013.01.01 00:00';
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    string   file_name;      // Variable für Speichern die Dateinamen
    string   filter="*.txt"; // Der Filter für die Suche nach Dateien
    datetime create_date;    // das Dateierstellungsdatum
    string   files[];        // Die Liste von Dateinamen
    int      def_size=25;     // die standardmässige Größe des Arrays
    int      size=0;         // die Anzahl der Dateien
//--- trennen Sie den Speicher für den Array ab
    ArrayResize(files,def_size);
//--- erhalten Sie die Suche Handle von Grund auf den lokaler Ordner
    long search_handle=FileFindFirst(filter,file_name);
//--- prüfen Sie, ob die FileFindFirst()Funktion erfolgreich durchgeführt hat
    if(search_handle!=INVALID_HANDLE)
    {
        //--- durchlaufen Sie im Zyklus Dateien
        do
        {
            files[size]=file_name;
            //--- erhöhen Sie die Größe des Arrays
            size++;
            if(size==def_size)
            {
                def_size+=25;
                ArrayResize(files,def_size);
            }
            //--- stürzen Sie den Fehlerwert
            ResetLastError();
            //--- haben Sie das Dateierstellungsdatum
            create_date=(datetime)FileGetInteger(file_name,FILE_CREATE_DATE,false);
            //--- prüfen Sie, ob die Datei alt ist
            if(create_date<InpFilesDate)
            {
                PrintFormat("Die %s Datei gelöscht!",file_name);
                //---löschen Sie die alte Datei
                FileDelete(file_name);
            }
        }
        while(FileFindNext(search_handle,file_name));
        //--- schließen Sie die Suche Handle
        FileFindClose(search_handle);
    }
    else
    {
        Print("Keine Dateien gefunden!");
        return;
    }
//--- prüfen Sie, ob welche Dateien blieben
    PrintFormat("Resultate:");
    for(int i=0;i<size;i++)
    {
        if(FileIsExist(files[i]))
            PrintFormat("Die %s Datei existiert!",files[i]);
        else

```

```
        PrintFormat("Die %s Datei gelöscht!",files[i]);  
    }  
}
```

Sehen Sie auch

[FileFindFirst](#)

## FileOpen

Funktion öffnet die Datei mit dem angegebenen Namen und angegebenen Flaggen.

```
int FileOpen(
    string file_name,           // der Dateiname
    int open_flags,            // die Kombination der Flaggen
    short delimiter='\t',      // der Begrenzer
    uint codepage=CP_ACP       // die Kodeseite
);
```

### Parameter

*file\_name*

[in] Name der geöffneten Datei, kann Unterordner haben. Wenn die Datei für Schreiben geöffnet wird, werden die angegebenen Unterordner erzeugt werden, wenn sie es nicht gibt.

*open\_flags*

[in] [Kombination der Flaggen](#), die Arbeitsmode mit Datei bestimmt. Flaggen werden folgenderweise bestimmt:

FILE\_READ Datei wird für Lesen geöffnet

FILE\_WRITE Datei wird für Schreiben geöffnet

FILE\_BIN Binärmode Lesen-Schreiben (ohne Umwandlung der Zeile in die Zeile)

FILE\_CSV Date des Typs csv (alle geschriebenen Elemente werden in die Zeilen des entsprechenden Typs umgewandelt, unicode oder ansi, und durch Begrenzer getrennt)

FILE\_TXT einfache Textdatei (dieselbe csv, aber Begrenzer wird nicht berücksichtigt)

FILE\_ANSI Zeilen des Typs ANSI (Einbyte-Symbole)

FILE\_UNICODE Zeilen des Typs UNICODE (Zweibyte-Symbole)

FILE\_SHARE\_READ Gesamter Lesezugang von mehreren Programmen

FILE\_SHARE\_WRITE Gesamter Schreibzugang von mehreren Programmen

FILE\_COMMON Lage der Datei im Gesamtordner aller Kundenterminals \Terminal\Common\Files

*delimiter='\t'*

[in] Wert, der als Begrenzer in txt oder csv-Datei verwendet wird. Wenn der Begrenzer für csv-Datei nicht angegeben wird, wird als Default-Wert Tab-Symbol verwendet. Wenn der Begrenzer für txt-Datei nicht angegeben wird, wird keiner Begrenzer verwendet. Wenn Begrenzer auf Null gestellt wird, wird keiner Begrenzer verwendet.

*codepage=CP\_ACP*

[in] Wert der Kodeseite. Für die am meisten verwendeten [Kodeseiten](#) werden entsprechende Konstanten vorausgesehen.

### Rückgabewert

Beim erfolgreichen Schließen gibt die Funktion der Datei Handle zurück, das danach für den Zugang zu Dateidaten verwendet wird). Beim Misserfolg gibt die Funktion [INVALID\\_HANDLE](#) zurück.

### Bemerkung

Aus Sicherheitsgründen ist die Arbeit mit Dateien in MQL5 streng gesteuert. Dateien, mit denen Datei-Operationen mittels MQL5 durchgeführt werden, können nicht außerhalb der Datei-Sandbox sein.

Datei wird im Ordner des Kundenterminals im Subordner MQL5\files geöffnet (oder Ordner\_des\_Testagents\MQL5\files beim Testen). Wenn unter Flaggen FILE\_COMMON angegeben wird, wird die Datei im Gesamtordner aller Kundenterminals MetaTrader5 geöffnet.

Man kann "die genannten Kanäle" nach den einhaltenden Regeln öffnen:

- Der Name des Kanals ist die Zeile, die aussehen soll: "\\servername\pipe\pipename", wo Servername ist der Name eines Servers im Netzwerk und Pipename ist der Name des Kanals. Wenn die Kanäle auf einem und derselbe Computer verwendet werden, kann der Name des Servers gesenkt sein, aber muss man anstelle seiner den Punkt machen: "\\.\pipe\pipename". Der Kunde, der versucht mit dem Kanal zu verbinden, soll seinen Namen wissen.
- Man muss [FileFlush\(\)](#) und [FileSeek\(\)](#) auf den Anfang der Datei zwischen aufeinander folgenden Operationen von lesen und schreiben aufgerufen werden.

In den Zeilen verwenden Sie einen umgekehrten Schrägstrich '\', also wenn Sie das Programm in MQL5 schreiben, Sie Backslash müssen '\' deshalb das obige Beispiel Ihren Code als "\\.\servername\pipe\pipename" schreiben.

Details zum Arbeiten mit die genannten Kanäle lesen Sie im Artikel ["Communicating With MetaTrader 5 Using Named Pipes Without Using DLLs"](#).

#### Beispiel:

```
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//--- unkorrektes Verfahren der Dateiuöffnung
string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
string filename=terminal_data_path+"\\MQL5\\Files\\"+"fractals.csv";
int filehandle=FileOpen(filename,FILE_WRITE|FILE_CSV);
if(filehandle<0)
{
Print("erfolgloser Versuch, die Datei durch absoluten Pfad zu uffnen");
Print("Fehlercode ",GetLastError());
};
//--- korrektes Verahren der Arbeit in "Sandkasten"
ResetLastError();
filehandle=FileOpen("fractals.csv",FILE_WRITE|FILE_CSV);
if(filehandle!=INVALID_HANDLE)
{
FileWrite(filehandle,TimeCurrent(),Symbol(),EnumToString(_Period));
FileClose(filehandle);
Print("FileOpen OK");
}
else Print("Operation FileOpen erfolglos, Fehler ",GetLastError());
//--- ein anderes Beispiel der Erzeugung des verschachtelten Verzeichnisses in MQL5\F
string subfolder="Research";
filehandle=FileOpen(subfolder+"\\fractals.txt",FILE_WRITE|FILE_CSV);
if(filehandle!=INVALID_HANDLE)
{
```

```
FileWrite(filehandle,TimeCurrent(),Symbol(), EnumToString(_Period));  
FileClose(filehandle);  
Print("Datei muss im Ordner "+terminal_data_path+"\\ "+subfolder+" erzeugt werden  
}  
else Print("Fehler beim Öffnen der Datei, Fehler ",GetLastError());  
; }
```

**Sehen Sie auch**

[Kodeseite Verwenden](#), [FileFindFirst](#), [FolderCreate](#), [Flaggen der Dateioffnung](#)

## FileClose

Schließen Sie die Datei, die früher durch die [FileOpen\(\)](#) Funktion geöffnet wurde.

```
void FileClose(  
    int file_handle    // Datei-Handle  
);
```

### Parameter

*file\_handle*

[in] Dateiattribut, das durch die FileOpen() Funktion zurückgegeben wird.

### Rückgabewert

Keinen Rückgabewert.

### Beispiel:

```
//--- Beim Starten des Scripts zeigen Sie den Fenster mit den Eingabeparametern an  
#property script_show_inputs  
//--- Eingabeparameters  
input string InpFileName="file.txt";    // der Dateiname  
input string InpDirectoryName="Data";    // der Verzeichnisname  
input int     InpEncodingType=FILE_ANSI; // ANSI=32 oder UNICODE=64  
//+-----+  
//| Script program start function |  
//+-----+  
void OnStart()  
{  
    //--- drucken Sie den Weg zum Folder in dem Sie arbeiten werden  
    PrintFormat("Arbeiten in dem %s\\Files\\ Folder ",TerminalInfoString(TERMINAL_DATA_...  
    //--- der Fehlerwert zu stürzen  
    ResetLastError();  
    //--- öffnen Sie die Datei für Lesen (wenn die Datei nicht existiert, tritt ein Fehler  
    int file_handle=FileOpen(InpDirectoryName+"\\"+InpFileName,FILE_READ|FILE_TXT|InpEn...  
    if(file_handle!=INVALID_HANDLE)  
    {  
        //--- drucken Sie den Dateiinhalte  
        while(!FileIsEnding(file_handle))  
            Print(FileReadString(file_handle));  
        //--- schließen Sie die Datei  
        FileClose(file_handle);  
    }  
    else  
        PrintFormat("Fehler, Kode = %d",GetLastError());  
}
```



## FileCopy

Kopiert Ausgangsdatei aus einem Lokale Ordner oder Gesamtordner in eine andere Datei.

```
bool FileCopy(  
    const string src_file_name,    // der Name der Quelle-Datei  
    int common_flag,              // die Lage  
    const string dst_file_name,    // der Name der Zieldatei  
    int mode_flags                 // der Zugangsmodus  
);
```

### Parameter

*src\_file\_name*

[in] Datename für Kopieren.

*common\_flag*

[in] [Flagge](#), die Dateilage bestimmt. Wenn `common_flag=FILE_COMMON`, befindet sich Datei im Gesamtordner aller Kundenterminals `\Terminal\Common\Files`. Anderenfalls befindet sich die Datei im Lokalen Ordner (zum Beispiel, `common_flag=0`).

*dst\_file\_name*

[in] Name der ergebenden Datei.

*mode\_flags*

[in] [Zugangsflaggen](#). Parameter kann erst 2 Flaggen enthalten: `FILE_REWRITE` und/oder `FILE_COMMON` - andere Flaggen werden ignoriert. Wenn Date existiert schon und dabei die Flagge `FILE_REWRITE` nicht angegeben wurde, wird die Datei nicht neugeschrieben und die Funktion gibt `false` zurück.

### Rückgabewert

Beim Misserfolg gibt die Funktion `false` zurück.

### Bemerkung

Aus Sicherheitsgründen ist die Arbeit mit Dateien in MQL5 streng gesteuert. Dateien, mit denen Datei-Operationen mittels MQL5 durchgeführt werden, können nicht außerhalb der Datei-Sandbox sein.

Wenn die neue Datei existiert, wird Kopieren abhängig davon durchgeführt, ob es die Flagge `FILE_REWRITE` im Wert des parameters gibt `mode_flags` oder nicht.

### Beispiel:

```

//--- Beim Starten des Scripts zeigen Sie den Fenster mit den Eingabeparametern an
#property script_show_inputs
//--- Eingabeparameters
input string InpSrc="source.txt";      // die Quelle
input string InpDst="destination.txt"; // Kopie
input int    InpEncodingType=FILE_ANSI; // ANSI=32 oder UNICODE=64
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//--- zeigen den Inhalt der Quelle an (er muss existieren sein)
if(!FileDisplay(InpSrc))
return;
//--- prüfen Sie, ob schon die Datei der Kopie existiert (es ist nicht verpflichtet, e
if(!FileDisplay(InpDst))
{
//--- die Datei der Kopie existiert nicht, das Kopieren ohne Fahne FILE_REWRITE
if(FileCopy(InpSrc,0,InpDst,0))
Print("Die Datei ist abgeschrieben!");
else
Print("Die Datei ist nicht abgeschrieben!");
}
else
{
//--- die Datei der Kopie existiert schon, wir werden versuchen, ohne FILE_REWRITE
if(FileCopy(InpSrc,0,InpDst,0))
Print("Die Datei ist abgeschrieben!");
else
Print("Die Datei ist nicht abgeschrieben!");
//--- der InpDst Dateiinhalt bleibt vorig
FileDisplay(InpDst);
//--- Wir werden noch einmal mit der FILE_REWRITE Fahne abschreiben (das richtig
if(FileCopy(InpSrc,0,InpDst,FILE_REWRITE))
Print("Die Datei ist abgeschrieben!");
else
Print("Die Datei ist nicht abgeschrieben!");
}
//--- haben die Kopie der Datei bekommen InpSrc
FileDisplay(InpDst);
}
//+-----+
//| Lesen des Inhalts einer Datei |
//+-----+
bool FileDisplay(const string file_name)
{
//--- der Fehlerwert zu stürzen
ResetLastError();
//--- öffnen Sie die Datei
int file_handle=FileOpen(file_name,FILE_READ|FILE_TXT|InpEncodingType);
if(file_handle!=INVALID_HANDLE)
{
//--- darstellen Sie den Dateiinhalt im Zyklus
Print("+-----+");
PrintFormat("der Dateiname = %s",file_name);
while(!FileIsEnding(file_handle))
Print(FileReadString(file_handle));
Print("+-----+");
//--- schließen Sie die Datei
FileClose(file_handle);
return(true);
}
}

```

```
    }  
    ///--- es ist damit gescheitert, die Datei zu öffnen  
    PrintFormat("%s ist nicht geöffnet, Fehler = %d",file_name,GetLastError());  
    return(false);  
}
```

## FileDelete

Entfernt die angegebene Datei im Lokalen Ordner des Kundenterminals.

```
bool FileDelete(  
    const string file_name,      // der Name der entfernten Datei  
    int common_flag=0           // die Lage der entfernten Datei  
);
```

### Parameter

*file\_name*

[in] Dateiname.

*common\_flag=0*

[in] [Flagge](#), die Dateilage bestimmt. Wenn `common_flag=FILE_COMMON`, befindet sich die Datei im Gesamtordner aller Kundenterminals `\Terminal\Common\Files`. Anderenfalls befindet sich die Datei im lokalen Ordner.

### Rückgabewert

Beim Misserfolg gibt die Funktion `false` zurück.

### Bemerkung

Aus Sicherheitsgründen ist die Arbeit mit Dateien in MQL5 streng gesteuert. Dateien, mit denen Datei-Operationen mittels MQL5 durchgeführt werden, können nicht außerhalb der Datei-Sandbox sein.

Entfernt die angegebene Datei im Lokalen Ordner des Kundenterminals (`MQL5\Files` oder `MQL5\Tester\Files` beim Testen). Wenn aber `common_flag=FILE_COMMON` angegeben wird, entfernt die Funktion die Datei aus dem Gesamtordner aller Kundenterminals.

### Beispiel:

```

//--- Beim Starten des Scripts zeigen Sie den Fenster mit den Eingabeparametern an
#property script_show_inputs
//---Datum für alte Dateien
input datetime InpFilesDate=D'2013.01.01 00:00';
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    string   file_name;      // Variable für Speichern die Dateinamen
    string   filter="*.txt"; // Der Filter für die Suche nach Dateien
    datetime create_date;    // das Dateierstellungsdatum
    string   files[];        // Die Liste von Dateinamen
    int      def_size=25;     // die standardmässige Größe des Arrays
    int      size=0;         // die Anzahl der Dateien
//--- trennen Sie den Speicher für den Array ab
    ArrayResize(files,def_size);
//--- erhalten Sie die Suche Handle von Grund auf den lokaler Ordner
    long search_handle=FileFindFirst(filter,file_name);
//--- prüfen Sie, ob die FileFindFirst() Funktion erfolgreich durchgeführt hat
    if(search_handle!=INVALID_HANDLE)
    {
        //--- durchlaufen Sie im Zyklus Dateien
        do
        {
            files[size]=file_name;
            //--- erhöhen Sie die Größe des Arrays
            size++;
            if(size==def_size)
            {
                def_size+=25;
                ArrayResize(files,def_size);
            }
            //--- der Fehlerwert zu stürzen
            ResetLastError();
            //--- haben Sie das Dateierstellungsdatum
            create_date=(datetime)FileGetInteger(file_name,FILE_CREATE_DATE,false);
            //--- prüfen Sie, ob die Datei alt ist
            if(create_date<InpFilesDate)
            {
                PrintFormat("Die %s Datei gelöscht!",file_name);
                //---löschen Sie die alte Datei
                FileDelete(file_name);
            }
        }
        while(FileFindNext(search_handle,file_name));
        //--- schließen Sie die Suche Handle
        FileFindClose(search_handle);
    }
    else
    {
        Print("Files not found!");
        return;
    }
//--- prüfen Sie, ob welche Dateien blieben
    PrintFormat("Resultate:");
    for(int i=0;i<size;i++)
    {
        if(FileIsExist(files[i]))
            PrintFormat("Die %s Datei existiert!",files[i]);
        else

```

```
        PrintFormat("Die %s Datei gelöscht!",files[i]);  
    }  
}
```

## FileMove

überträgt die Datei aus einem lokalen oder Gesamtordner in einen anderen Ordner.

```
bool FileMove(  
    const string src_file_name,    // der Dateiname für Übertragung  
    int common_flag,              // die Lage  
    const string dst_file_name,    // der Name der Zielformat  
    int mode_flags                // Zugangsverfahren  
);
```

### Parameter

*src\_file\_name*

[in] Dateiname für Übertragung/Umbenennung.

*common\_flag*

[in] [Flagge](#), die die Lage der Datei bestimmt. Wenn `common_flag=FILE_COMMON` ist, befindet sich die Datei im Gesamtordner aller Kundenterminals `\Terminal\Common\Files`. Anderenfalls befindet sich die Datei im lokalen Ordner (`common_flag=0`).

*dst\_file\_name*

[in] Name der ergebenden Datei.

*mode\_flags*

[in] [Zugangsflaggen](#). Parameter kann nur zwei Flaggen enthalten: `FILE_REWRITE` und/oder `FILE_COMMON` - andere Flaggen werden ignoriert. Wenn die Datei existiert schon und dabei die Flagge `FILE_REWRITE` nicht angegeben wurde, wird die Datei nicht neugeschrieben, und die Funktion gibt `false` zurück.

### Rückgabewert

Beim Misserfolg gibt die Funktion `false` zurück.

### Bemerkung

Aus Sicherheitsgründen ist die Arbeit mit Dateien in MQL5 streng gesteuert. Dateien, mit denen Datei-Operationen mittels MQL5 durchgeführt werden, können nicht außerhalb der Datei-Sandbox sein.

Wenn die neue Datei existiert schon, wird das Kopieren abhängig davon durchgeführt werden, ob die Flagge `FILE_REWRITE` im Parameterwert `mode_flags` gibt.

### Beispiel:

```

//--- Beim Starten des Scripts zeigen Sie den Fenster mit den Eingabeparametern an
#property script_show_inputs
//--- Eingabeparameters
input string InpSrcName="data.txt";
input string InpDstName="newdata.txt";
input string InpSrcDirectory="SomeFolder";
input string InpDstDirectory="OtherFolder";
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    string local=TerminalInfoString(TERMINAL_DATA_PATH);
    string common=TerminalInfoString(TERMINAL_COMMONDATA_PATH);
//--- erhalten Sie die Dateipfade
    string src_path;
    string dst_path;
    StringConcatenate(src_path,InpSrcDirectory,"\\",InpSrcName);
    StringConcatenate(dst_path,InpDstDirectory,"\\",InpDstName);
//--- prüfen Sie, ob die Quelle Datei existiert (wenn es gibt keine - Ausgabe)
    if(FileExists(src_path))
        PrintFormat("%s Datei existiert in der %s\\Files\\%s Ordner nicht",InpSrcName,local);
    else
    {
        PrintFormat("Fehler, %s Hauptdatei nicht gefunden",InpSrcName);
        return;
    }
//--- prüfen Sie, ob die Ergebnisdatei existiert
    if(FileExists(dst_path,FILE_COMMON))
    {
        PrintFormat("%s Datei existiert in der %s\\Files\\%s Ordner nicht",InpDstName,common);
//--- die Datei existiert, die Umstellung muss man mit der Fahne FILE_REWRITE durchführen
        ResetLastError();
        if(FileMove(src_path,0,dst_path,FILE_COMMON|FILE_REWRITE))
            PrintFormat("%s Datei ist versetzt",InpSrcName);
        else
            PrintFormat("Fehler! Kode = %d",GetLastError());
    }
    else
    {
        PrintFormat("%s Datei existiert in der %s\\Files\\%s Ordner nicht",InpDstName,common);
//--- die Datei existiert nicht, die Umstellung muss man ohne der Fahne FILE_REWRITE durchführen
        ResetLastError();
        if(FileMove(src_path,0,dst_path,FILE_COMMON))
            PrintFormat("%s Datei ist versetzt",InpSrcName);
        else
            PrintFormat("Fehler! Kode = %d",GetLastError());
    }
//--- jetzt ist die Datei versetzt, überprüfen Sie es
    if(FileExists(dst_path,FILE_COMMON) && !FileExists(src_path,0))
        Print("Erfolgreich!");
    else
        Print("Fehler!");
}

```

Sehen Sie auch

[FileExists](#)



## FileFlush

Speichert in der Platte alle Daten, die im Dateipuffer Eingabe-Ausgabe noch bleiben.

```
void FileFlush(  
    int file_handle    // Datei-Handle  
);
```

### Parameter

*file\_handle*

[in] Dateiattribut, das durch die [FileOpen\(\)](#) Funktion zurückgegeben wird.

### Rückgabewert

Keinen Rückgabewert.

### Bemerkung

Wenn Sie in eine Datei schreiben, können die Daten physisch nur nach einiger Zeit sein. Damit die Daten sofort in die Datei erhalten, müssen Sie die FileFlush() Funktion verwenden. Wenn Sie nicht der Daten-Funktion verwenden, die auf den Datenträger nicht noch aufgeholt hat gibt es nur, wenn Sie die FileClose() Funktion schließen.

Die Funktion sollte verwendet werden, wenn die Aufnahmedaten einige Wert darstellt. Es sollte berücksichtigt werden, dass ein häufige Funktionsaufruf die Geschwindigkeit des Programms beeinflussen kann.

Die FileFlush() Funktion muss zwischen der Operationen des Lesens aus der Datei und Schreibens in die Datei aufgerufen werden.

### Beispiel:

```

//--- Beim Starten des Scripts zeigen Sie den Fenster mit den Eingabeparametern an
#property script_show_inputs
//--- der Name der Datei zum Schreiben
input string InpFileName="example.csv"; // der Dateiname
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//--- Stürzen Sie den Fehlerwert
ResetLastError();
//--- Öffnen Sie die Datei
int file_handle=FileOpen(InpFileName,FILE_READ|FILE_WRITE|FILE_CSV);
if(file_handle!=INVALID_HANDLE)
{
//--- Daten in eine Datei schreiben
for(int i=0;i<1000;i++)
{
//--- Aufnahmefunktion aufrufen
FileWrite(file_handle,TimeCurrent(),SymbolInfoDouble(Symbol(),SYMBOL_BID),Sym
//--- Zurücksetzen der Daten auf der Festplatte bei jeder 128 Iteration
if((i & 127)==127)
{
//---Nun wird die Daten in der Datei, und wenn das Terminal wir schwerwie
FileFlush(file_handle);
}
//---die Verzögerung von 0,01 Sekunden
Sleep(10);
}
//--- Schließen Sie die Datei
FileClose(file_handle);
}
else
PrintFormat("Fehler, Kode = %d",GetLastError());
}

```

Sehen Sie auch

[FileClose](#)

## FileGetInteger

Ruft eine ganzzahlige Eigenschaft der Datei. Es gibt 2 Varianten der Funktion.

1. Eigenschaften beim Handle bekommen.

```
long FileGetInteger(  
    int file_handle, // Datei-Handle  
    ENUM_FILE_PROPERTY_INTEGER property_id // ID der Eigenschaft  
);
```

2. Eigenschaften beim Dateinamen bekommen.

```
long FileGetInteger(  
    const string file_name, // der Dateiname  
    ENUM_FILE_PROPERTY_INTEGER property_id, // ID der Eigenschaft  
    bool common_folder=false // Die Datei ist in einem lokalen Ordner  
); // oder in einem freigegebenen Ordner
```

### Parameter

*file\_handle*

[in] Dateideskriptor der durch die [FileOpen\(\)](#) Funktion zurückgegeben wird.

*file\_name*

[in] Der Dateiname.

*property\_id*

[in] ID der Datei-Eigenschaft. Der Wert kann der [ENUM\\_FILE\\_PROPERTY\\_INTEGER](#) Enumerationswerte sein. Wenn die zweite Version der Funktion benutzt wird, können Sie die Werte nur für die [folgenden Eigenschaften](#) erhalten: FILE\_EXISTS, FILE\_CREATE\_DATE, FILE\_MODIFY\_DATE, FILE\_ACCESS\_DATE und FILE\_SIZE.

*common\_folder=false*

[in] Verweist auf den Speicherort der Datei. Wenn der Parameter falsch ist, dann das Data-Ordner des Terminals durchgeschaut wird, ansonsten wird es davon ausgegangen, dass die Datei in einem freigegebenen Ordner von Kunden-Terminals liegt \Terminal\Common\Files ([FILE\\_COMMON](#)).

### Rückgabewert

Wert der Eigenschaft. Im Falle eines Fehlers, gibt die Funktion -1 zurück. Um die Information über [Fehler](#) zu erhalten, muss man die Funktion [GetLastError\(\)](#) aufrufen.

Wenn ein Ordner angegeben wird, wenn Sie die Eigenschaften durch den Namen erhalten, hat die Funktion Fehler 5018 (ERR\_MQL\_FILE\_IS\_DIRECTORY) in jedem Fall, während der Rückgabewert korrekt ist.

### Bemerkung

Die Funktion verändert immer den Fehlercode. Wenn es gelingt, wird der Fehlercode auf Null zurückgesetzt.

### Beispiel:

```

//--- Beim Starten des Scripts zeigen Sie den Fenster mit den Eingabeparametern an
#property script_show_inputs
//--- Eingabeparameters
input string InpFileName="data.csv";
input string InpDirectoryName="SomeFolder";
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    string path=InpDirectoryName+"\\"+InpFileName;
    long l=0;
//--- Öffnen Sie die Datei
    ResetLastError();
    int handle=FileOpen(path,FILE_READ|FILE_CSV);
    if(handle!=INVALID_HANDLE)
    {
        //--- drucken Sie alle Informationen über die Datei
        Print(InpFileName," file info:");
        FileInfo(handle,FILE_EXISTS,l,"bool");
        FileInfo(handle,FILE_CREATE_DATE,l,"date");
        FileInfo(handle,FILE_MODIFY_DATE,l,"date");
        FileInfo(handle,FILE_ACCESS_DATE,l,"date");
        FileInfo(handle,FILE_SIZE,l,"other");
        FileInfo(handle,FILE_POSITION,l,"other");
        FileInfo(handle,FILE_END,l,"bool");
        FileInfo(handle,FILE_IS_COMMON,l,"bool");
        FileInfo(handle,FILE_IS_TEXT,l,"bool");
        FileInfo(handle,FILE_IS_BINARY,l,"bool");
        FileInfo(handle,FILE_IS_CSV,l,"bool");
        FileInfo(handle,FILE_IS_ANSI,l,"bool");
        FileInfo(handle,FILE_IS_READABLE,l,"bool");
        FileInfo(handle,FILE_IS_WRITABLE,l,"bool");
        //--- schließen Sie die Datei
        FileClose(handle);
    }
    else
        PrintFormat("%s Datei ist nicht geöffnet, FehlerCode = %d",InpFileName,GetLastError());
}
//+-----+
//| Anzeige den Wert der Dateieigenschaft |
//+-----+
void FileInfo(const int handle,const ENUM_FILE_PROPERTY_INTEGER id,
             long l,const string type)
{
    //--- erhalten Sie den Wert der Eigenschaft
    ResetLastError();
    if((l=FileGetInteger(handle,id))!=-1)
    {
        //--- der Wert erhalten wurde, zeigen Sie es im richtigen Format
        if(!StringCompare(type,"bool"))
            Print(EnumToString(id)," = ",l ? "true" : "false");
        if(!StringCompare(type,"date"))
            Print(EnumToString(id)," = ",(datetime)l);
        if(!StringCompare(type,"other"))
            Print(EnumToString(id)," = ",l);
    }
    else
        Print("Fehler, Kode = ",GetLastError());
}

```

Sehen Sie auch

[Dateioperationen](#), [Dateieigenschaften](#)

## FileIsEnding

Bestimmt Dateiende während des Lesens.

```
bool FileIsEnding(
    int file_handle    // Datei-Handle
);
```

### Parameter

*file\_handle*

[in] Dateiattribut, das durch die [FileOpen\(\)](#) Funktion zurückgegeben wird.

### Rückgabewert

Funktion gibt true zurück, wenn während des Lesens oder Übertragung des Dateianzeigers das Dateiende erreicht wurde.

### Bemerkung

Um das Dateiende zu ermitteln, versucht die Funktion lesen auf die nächste Zeile aus der Datei lesen. Wenn es nicht vorhanden ist, gibt die Funktion true, andernfalls false.

### Beispiel:

```

//--- Beim Starten des Scripts zeigen Sie den Fenster mit den Eingabeparametern an
#property script_show_inputs
//--- Eingabeparameters
input string InpFileName="file.txt";    // der Dateiname
input string InpDirectoryName="Data";    // der Verzeichnisname
input int    InpEncodingType=FILE_ANSI; // ANSI=32 oder UNICODE=64
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    //--- drucken Sie den Weg zum Folder in dem Sie arbeiten werden
    PrintFormat("Arbeiten in dem %s\\Files\\ Ordner ",TerminalInfoString(TERMINAL_DATA_
//--- der Fehlerwert zu stürzen
    ResetLastError();
//--- öffnen Sie die Datei für Lesen (wenn die Datei nicht existiert, tritt ein Fehler
    int file_handle=FileOpen(InpDirectoryName+"\\\\"+InpFileName,FILE_READ|FILE_TXT|InpEn
    if(file_handle!=INVALID_HANDLE)
    {
        //--- drucken Sie den Dateiinhalte
        while(!FileIsEnding(file_handle))
            Print(FileReadString(file_handle));
        //--- schließen Sie die Datei
        FileClose(file_handle);
    }
    else
        PrintFormat("Fehler, Kode = %d",GetLastError());
}
```

## FileIsLineEnding

Bestimmt Zeilenende während des Lesens.

```
bool FileIsLineEnding(  
    int file_handle    // Datei-Handle  
);
```

### Parameter

*file\_handle*

[in] Dateiattribut, das durch die Funktion [FileOpen\(\)](#) zurückgegeben wird.

### Rückgabewert

Gibt true zurück wenn beim Lesen der txt oder csv-Datei das Zeilenende erreicht wird (Symbole CR-LF).

**Beispiel** (die Datei wird verwendet, die infolge der Arbeit des Beispiels für die Funktion [FileWriteString](#) bekommen ist)

```

//+-----+
//|                                     Demo_FileIsLineEnding.mq5 |
//|                                     Copyright 2013, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots  1
//---- plot Label1
#property indicator_label1  "Overbought & Oversold"
#property indicator_type1   DRAW_COLOR_BARS
#property indicator_color1  clrRed, clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
//--- Parameter zum Lesen von Daten
input string InpFileName="RSI.csv"; // der Dateiname
input string InpDirectoryName="Data"; // der Verzeichnisname
//--- die Indikator-Buffers
double   open_buff[];
double   high_buff[];
double   low_buff[];
double   close_buff[];
double   color_buff[];
//--- überkauft Variablen
int       ovb_ind=0;
int       ovb_size=0;
datetime ovb_time[];
//--- überverkauft Variablen
int       ovs_ind=0;
int       ovs_size=0;
datetime ovs_time[];
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- die standartmäßige Variable Größe von Arrays
int ovb_def_size=100;
int ovs_def_size=100;
//--- trennen Sie den Speichern für den Arrays
ArrayResize(ovb_time,ovb_def_size);
ArrayResize(ovs_time,ovs_def_size);
//--- öffnen Sie die Datei
ResetLastError();
int file_handle=FileOpen(InpDirectoryName+"\\"+InpFileName,FILE_READ|FILE_CSV|FILE_
if(file_handle!=INVALID_HANDLE)
{
PrintFormat(" %s Datei ist zum Lesen geöffnet",InpFileName);
PrintFormat("Pfad zur Datei: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH)
double value;
//--- lesen Sie die Dateidaten
while(!FileIsEnding(file_handle))
{
//--- lesen Sie den ersten Wert in eine Zeichenfolge
value=FileReadNumber(file_handle);
//--- lesen Sie die verschiedenen Arrays, abhängig vom Ergebnis der Funktion
if(value>=70)
ReadData(file_handle,ovb_time,ovb_size,ovb_def_size);

```



```

        else
            ReadData(file_handle, ovs_time, ovs_size, ovs_def_size);
    }
    //--- schließen Sie die Datei
    FileClose(file_handle);
    PrintFormat("Die Daten werden gelesen, die Datei %s geschlossen", InpFileName);
}
else
{
    PrintFormat("Fehler beim Öffnen der Datei %s, Fehlercode = %d", InpFileName, GetLastError());
    return(INIT_FAILED);
}
}
//--- Arrays-Bindung
SetIndexBuffer(0, open_buff, INDICATOR_DATA);
SetIndexBuffer(1, high_buff, INDICATOR_DATA);
SetIndexBuffer(2, low_buff, INDICATOR_DATA);
SetIndexBuffer(3, close_buff, INDICATOR_DATA);
SetIndexBuffer(4, color_buff, INDICATOR_COLOR_INDEX);
//---- erstellen Sie die Anzeige-Werte, die im Diagramm nicht sichtbar sein werden
PlotIndexSetDouble(0, PLOT_EMPTY_VALUE, 0);
//---
return(INIT_SUCCEEDED);
}
//+-----+
//| Lesen Sie die Zeilen einer Datei |
//+-----+
void ReadData(const int file_handle, datetime &arr[], int &size, int &def_size)
{
    bool flag=false;
    //--- Lesen Sie, bis das Ende der Zeichenfolge oder Datei erreichen
    while(!FileIsLineEnding(file_handle) && !FileIsEnding(file_handle))
    {
        //--- bewegen Sie den Wagen, nach dem Lesen der Anzahl
        if(flag)
            FileReadNumber(file_handle);
        //--- das aktuelle Datum speichern
        arr[size]=FileReadDatetime(file_handle);
        size++;
        //--- Falls erforderlich, vergrößern Sie die Größe des Arrays
        if(size==def_size)
        {
            def_size+=100;
            ArrayResize(arr, def_size);
        }
        //--- übergeben die erste iteration
        flag=true;
    }
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])

```

```
{
```

```

    ArraySetAsSeries(time, false);
    ArraySetAsSeries(open, false);
    ArraySetAsSeries(high, false);
    ArraySetAsSeries(low, false);
    ArraySetAsSeries(close, false);
    ///--- der Zyklus für die noch unbearbeiteten Bars
    for(int i=prev_calculated; i<rates_total; i++)
    {
        ///--- die Voreinstellung 0
        open_buff[i]=0;
        high_buff[i]=0;
        low_buff[i]=0;
        close_buff[i]=0;
        color_buff[i]=0;
        ///--- prüfen Sie, ob es noch Daten gibt
        if(ovb_ind<ovb_size)
            for(int j=ovb_ind; j<ovb_size; j++)
            {
                ///--- wenn die Daten identisch sind, ist die Bar in überkauft Zone
                if(time[i]==ovb_time[j])
                {
                    open_buff[i]=open[i];
                    high_buff[i]=high[i];
                    low_buff[i]=low[i];
                    close_buff[i]=close[i];
                    ///--- 0 - die rote Farbe
                    color_buff[i]=0;
                    ///--- vergrößern Sie den Zähler
                    ovb_ind=j+1;
                    break;
                }
            }
        ///--- prüfen Sie, ob es noch Daten gibt
        if(ovs_ind<ovs_size)
            for(int j=ovs_ind; j<ovs_size; j++)
            {
                ///--- wenn die Daten identisch sind, ist die Bar in überverkauft Zone
                if(time[i]==ovs_time[j])
                {
                    open_buff[i]=open[i];
                    high_buff[i]=high[i];
                    low_buff[i]=low[i];
                    close_buff[i]=close[i];
                    ///--- 1 - die blau Farbe
                    color_buff[i]=1;
                    ///--- vergrößern Sie den Zähler
                    ovs_ind=j+1;
                    break;
                }
            }
    }
    ///--- den Wert prev_calculated für den nächsten Anruf zurückgeben
    return(rates_total);
}
//+-----+
//| Der Bearbeiter des Ereignisses ChartEvent |
//+-----+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam

```

```
    )  
  
    {  
    //--- ändern Sie die Breite des Indikators je nach dem Ausmaß  
    if (ChartGetInteger (0, CHART_SCALE) > 3)  
        PlotIndexSetInteger (0, PLOT_LINE_WIDTH, 2);  
    else  
        PlotIndexSetInteger (0, PLOT_LINE_WIDTH, 1);  
    }
```

Sehen Sie auch

[FileWriteString](#)

## FileReadArray

Liest Felder aller Typen ausser Zeilentypen (kann Feld der Strukturen ohne Zeilen und dynamische Felder), aus Binärdatei von der laufenden Position des Dateianzeigers.

```
uint FileReadArray(  
    int file_handle,           // Datei-Handle  
    void& array[],            // Feld für Aufzeichnung  
    int start=0,              // Anfangsposition für Schreiben  
    int count=WHOLE_ARRAY     // Anzahl für Lesen  
);
```

### Parameter

*file\_handle*

[in] Dateiattribut, das durch die Funktion [FileOpen\(\)](#) zurückgegeben wird.

*array[]*

[out] Feld, wohin Daten geladen werden.

*start=0*

[in] Anfangsposition für Schreiben in das Feld.

*count=WHOLE\_ARRAY*

[in] Anzahl der Elemente für Lesen.

### Rückgabewert

Anzahl der gelesenen Elemente. Als Default liest das ganze Feld (count=[WHOLE\\_ARRAY](#)).

### Bemerkung

Zeilenfeld kann nur aus Datei des Typs TXT gelesen werden. Wenn notwendig versucht die Funktion die Größe des Feldes zu vergrößern.

**Beispiel** (die Datei wird verwendet, die infolge der Arbeit des Beispiels für die Funktion [FileWriteArray](#) bekommen ist)

```

//--- Beim Starten des Scripts zeigen Sie den Fenster mit den Eingabeparametern an
#property script_show_inputs
//--- Eingabeparameters
input string InpFileName="data.bin";
input string InpDirectoryName="SomeFolder";
//+-----+
//| Struktur zum Speichern von Daten auf die Preise |
//+-----+
struct prices
{
    datetime      date; // Datum
    double        bid;  // der Bid Preis
    double        ask;  // der Ask Preis
};
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    //--- ein Array von Struktur
    prices arr[];
    //--- Pfad zur Datei
    string path=InpDirectoryName+"\\"+InpFileName;
    //--- öffnen Sie die Datei
    ResetLastError();
    int file_handle=FileOpen(path,FILE_READ|FILE_BIN);
    if(file_handle!=INVALID_HANDLE)
    {
        //--- lesen Sie alle Dateteidaten in ein Array
        FileReadArray(file_handle,arr);
        //--- erhalten Sie die Größe des Arrays
        int size=ArraySize(arr);
        //--- drucken Sie die Daten aus dem Array
        for(int i=0;i<size;i++)
            Print("Daten = ",arr[i].date," Bid = ",arr[i].bid," Ask = ",arr[i].ask);
        Print("Gesamtdaten = ",size);
        //--- schließen Sie die Datei
        FileClose(file_handle);
    }
    else
        Print("Fehler beim Öffnen der Datei, Fehler ",GetLastError());
}

```

Sehen Sie auch

[Variablen](#), [FileWriteArray](#)

## FileReadBool

Liest aus der Datei des Typs CSV die Zeile von der laufenden Position bis zum Begrenzer (oder bis zum Ende der Textzeile) und wandelt die gelesene Zeile in den Wert des Typs bool um.

```
bool FileReadBool(  
    int file_handle    // Datei-Handle  
);
```

### Parameter

*file\_handle*

[in] Dateiattribut, das durch die Funktion [FileOpen\(\)](#).

### Rückgabewert

Die gelesene Zeile kann die Werte "true", "false" oder Symboldarstellung der Ganzzahlen "0" oder "1" haben. Der Nichtnullwert wird in den logischen Wert true umgewandelt. Die Funktion gibt den erhaltenen umgewandelten Wert zurück.

**Beispiel** (die Datei wird verwendet, die infolge der Arbeit des Beispiels für die Funktion [FileWrite](#) bekommen ist)

```

//+-----+
//|                                     Demo_FileReadBool.mq5 |
//|                                     Copyright 2013, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots  2
//---- plot Label1
#property indicator_label1  "UpSignal"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  4
//---- plot Label2
#property indicator_label2  "DownSignal"
#property indicator_type2   DRAW_ARROW
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  4
//--- Parameter zum Lesen von Daten
input string InpFileName="MACD.csv"; // der Dateiname
input string InpDirectoryName="Data"; // der Verzeichnisname
//--- globale Variablen
int      ind=0;           // Index
double   upbuff[];       // der Indikator-Puffer der Zeiger nach oben
double   downbuff[];     // der Indikator-Puffer der Zeiger nach unten
bool     sign_buff[];     // ein Array von Signalen (true - kaufen, false - verkaufen)
datetime time_buff[];    // ein Array der Zeit des Eintritts der Signale
int      size=0;         // die Größe der Array von Signalen
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- öffnen Sie die Datei
ResetLastError();
int file_handle=FileOpen(InpDirectoryName+"\\"+InpFileName,FILE_READ|FILE_CSV);
if(file_handle!=INVALID_HANDLE)
{
PrintFormat("%s Datei ist zum Lesen geöffnet",InpFileName);
//--- Lesen Sie zuerst die Anzahl der Signale
size=(int)FileReadNumber(file_handle);
//--- trennen Sie den Speichern für den Arrays
ArrayResize(sign_buff,size);
ArrayResize(time_buff,size);
//--- lesen Sie die Dateidaten
for(int i=0;i<size;i++)
{
//--- Die Zeit des Signals
time_buff[i]=FileReadDatetime(file_handle);
//--- Der Signalwert
sign_buff[i]=FileReadBool(file_handle);
}
//--- schließen Sie die Datei
FileClose(file_handle);
}
else

```



```

    {
        PrintFormat("Fehler beim Öffnen der Datei %s, Fehlercode = %d",InpFileName,GetLastError());
        return(INIT_FAILED);
    }
}

//--- Arrays-Bindung
SetIndexBuffer(0,upbuff,INDICATOR_DATA);
SetIndexBuffer(1,downbuff,INDICATOR_DATA);
//--- geben Sie den Zeichencode für das Rendering in PLOT_ARROW vor
PlotIndexSetInteger(0,PLOT_ARROW,241);
PlotIndexSetInteger(1,PLOT_ARROW,242);
//---- erstellen Sie die Indikatorwerte, die im Diagramm nicht sichtbar sein werden
PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
PlotIndexSetDouble(1,PLOT_EMPTY_VALUE,0);
//---
return(INIT_SUCCEEDED);
}

//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{

```

```

    ArraySetAsSeries(time, false);
    ArraySetAsSeries(high, false);
    ArraySetAsSeries(low, false);
    //--- der Zyklus für die noch unbearbeiteten Bars
    for(int i=prev_calculated; i<rates_total; i++)
    {
        //--- die Voreinstellung 0
        upbuff[i]=0;
        downbuff[i]=0;
        //--- prüfen Sie, ob es noch Daten gibt
        if(ind<size)
        {
            for(int j=ind; j<size; j++)
            {
                //--- Wenn die Daten übereinstimmen, dann verwenden Sie den Wert aus einer
                if(time[i]==time_buff[j])
                {
                    //--- zeichnen Sie den Zeiger je nach dem Signal
                    if(sign_buff[j])
                        upbuff[i]=high[i];
                    else
                        downbuff[i]=low[i];
                    //--- vergrößern Sie den Zähler
                    ind=j+1;
                    break;
                }
            }
        }
    }
    //--- den Wert prev_calculated für den nächsten Anruf zurückgeben
    return(rates_total);
}

```

Sehen Sie auch

[Typ bool](#), [FileWrite](#)

## FileReadDatetime

Liest von der Datei des Typs CSV eine Zeile eines der Formate: "YYYY.MM.DD HH:MI:SS", "YYYY.MM.DD" oder "HH:MI:SS" - und wandelt sie in den Wert des Typs datetime um.

```
datetime FileReadDatetime(  
    int file_handle    // Datei-Handle  
);
```

### Parameter

*file\_handle*

[in] Dateiattribut, das durch die Funktion [FileOpen\(\)](#) zurückgegeben wird.

### Rückgabewert

Wert des Typs datetime.

**Beispiel** (die Datei wird verwendet, die infolge der Arbeit des Beispiels für die Funktion [FileWrite](#) bekommen ist)

```

//+-----+
//|                                     Demo_FileReadDateTime.mq5 |
//|                                     Copyright 2013, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots  2
//---- plot Label1
#property indicator_label1  "UpSignal"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  4
//---- plot Label2
#property indicator_label2  "DownSignal"
#property indicator_type2   DRAW_ARROW
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  4
//--- Parameter zum Lesen von Daten
input string InpFileName="MACD.csv"; // der Dateiname
input string InpDirectoryName="Data"; // der Verzeichnisname
//--- globale Variablen
int      ind=0;           // Index
double   upbuff[];       // der Indikator-Puffer der Zeiger nach oben
double   downbuff[];     // der Indikator-Puffer der Zeiger nach unten
bool     sign_buff[];     // ein Array von Signalen (true - kaufen, false - verkaufen)
datetime time_buff[];    // ein Array der Zeit des Eintritts der Signale
int      size=0;         // die Größe der Array von Signalen
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- öffnen Sie die Datei
ResetLastError();
int file_handle=FileOpen(InpDirectoryName+"\\"+InpFileName,FILE_READ|FILE_CSV);
if(file_handle!=INVALID_HANDLE)
{
PrintFormat("%s Datei ist zum Lesen geöffnet",InpFileName);
//--- Lesen Sie zuerst die Anzahl der Signale
size=(int)FileReadNumber(file_handle);
//--- trennen Sie den Speichern für den Arrays
ArrayResize(sign_buff,size);
ArrayResize(time_buff,size);
//--- lesen Sie die Dateidaten
for(int i=0;i<size;i++)
{
//--- Die Zeit des Signals
time_buff[i]=FileReadDatetime(file_handle);
//--- Der Signalwert
sign_buff[i]=FileReadBool(file_handle);
}
//--- schließen Sie die Datei
FileClose(file_handle);
}
else

```

```

    {
        PrintFormat("Fehler beim Öffnen der Datei %s, Fehlercode = %d",InpFileName,GetLastError());
        return(INIT_FAILED);
    }
}

//--- Arrays-Bindung
SetIndexBuffer(0,upbuff,INDICATOR_DATA);
SetIndexBuffer(1,downbuff,INDICATOR_DATA);
//--- geben Sie den Zeichencode für das Rendering in PLOT_ARROW vor
PlotIndexSetInteger(0,PLOT_ARROW,241);
PlotIndexSetInteger(1,PLOT_ARROW,242);
//---- erstellen Sie die Indikatorwerte, die im Diagramm nicht sichtbar sein werden
PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
PlotIndexSetDouble(1,PLOT_EMPTY_VALUE,0);
//---
return(INIT_SUCCEEDED);
}

//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{

```

```
ArraySetAsSeries(time, false);
ArraySetAsSeries(high, false);
ArraySetAsSeries(low, false);
//--- der Zyklus für die noch unbearbeiteten Bars
for(int i=prev_calculated;i<rates_total;i++)
{
    //--- die Voreinstellung 0
    upbuff[i]=0;
    downbuff[i]=0;
    //--- prüfen Sie, ob es noch Daten gibt
    if(ind<size)
    {
        for(int j=ind;j<size;j++)
        {
            //--- Wenn die Daten übereinstimmen, dann verwenden Sie den Wert aus einer
            if(time[i]==time_buff[j])
            {
                //--- zeichnen Sie den Zeiger je nach dem Signal
                if(sign_buff[j])
                    upbuff[i]=high[i];
                else
                    downbuff[i]=low[i];
                //--- vergrößern Sie den Zähler
                ind=j+1;
                break;
            }
        }
    }
}
//--- den Wert prev_calculated für den nächsten Anruf zurückgeben
return(rates_total);
}
```

Sehen Sie auch

[Typ datetime](#), [StringToTime](#), [TimeToString](#), [FileWrite](#)

## FileReadDouble

Liest die Zahl der doppelten Genauigkeit mit dem Fließpunkt (double) aus der Binärdatei von der laufenden Position des Dateianzeigers.

```
double FileReadDouble(  
    int file_handle    // Datei-Handle  
);
```

### Parameter

*file\_handle*

[in] Dateiattribut, das durch die Funktion [FileOpen\(\)](#) zurückgegeben wird.

### Rückgabewert

Wert des Typs double.

### Bemerkung

Für die Erhaltung der fehlerbezogenen Information muss die Funktion [GetLastError\(\)](#) aufgerufen werden.

**Beispiel** (die Datei wird verwendet, die infolge der Arbeit des Beispiels für die Funktion [FileWriteDouble](#) bekommen ist)

```
//+-----+
//|                                     Demo_FileReadDouble.mq5 |
//|                                     Copyright 2013, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots  1
//---- plot Label1
#property indicator_label1  "MA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
#property indicator_separate_window
//---- Parameter zum Lesen von Daten
input string InpFileName="MA.csv";    // der Dateiname
input string InpDirectoryName="Data"; // der Verzeichnisname
//--- globale Variablen
int      ind=0;
int      size=0;
double   ma_buff[];
datetime time_buff[];
//--- Indikator Buffer
double   buff[];
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- öffnen Sie die Datei
ResetLastError();
int file_handle=FileOpen(InpDirectoryName+"\\"+InpFileName,FILE_READ|FILE_BIN);
if(file_handle!=INVALID_HANDLE)
{
PrintFormat("%s Datei ist zum Lesen geöffnet",InpFileName);
PrintFormat("Pfad zur Datei: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
//--- lesen Sie zuerst wie viele Daten im Datei sind
size=(int)FileReadDouble(file_handle);
//--- trennen Sie den Speichern für den Arrays
ArrayResize(ma_buff,size);
ArrayResize(time_buff,size);
//--- lesen Sie den Dateidaten
for(int i=0;i<size;i++)
{
time_buff[i]=(datetime)FileReadDouble(file_handle);
ma_buff[i]=FileReadDouble(file_handle);
}
//--- schließen Sie die Datei
FileClose(file_handle);
PrintFormat("Die Daten werden gelesen, %s Datei geschlossen",InpFileName);
}
else
{
PrintFormat("Fehler beim Öffnen der Datei %s, Fehlercode = %d",InpFileName,GetLastError());
return(INIT_FAILED);
}
}
//--- Array-Bindung zum Indikator Puffer mit 0 Index
```



```

    SetIndexBuffer(0, buff, INDICATOR_DATA);
    //--- erstellen Sie die Indikatorwerte, die im Diagramm nicht sichtbar sein werden
    PlotIndexSetDouble(0, PLOT_EMPTY_VALUE, 0);
    //---
    return(INIT_SUCCEEDED);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
    ArraySetAsSeries(time, false);
    //--- der Zyklus für noch unbearbeiteten Bars
    for(int i=prev_calculated; i<rates_total; i++)
    {
        //--- die Voreinstellung 0
        buff[i]=0;
        //--- prüfen Sie, ob es noch Daten gibt
        if(ind<size)
        {
            for(int j=ind; j<size; j++)
            {
                //--- wenn die Daten identisch sind, dann verwenden Sie den Wert aus einer
                if(time[i]==time_buff[j])
                {
                    buff[i]=ma_buff[j];
                    //--- vergrößern Sie den Zähler
                    ind=j+1;
                    break;
                }
            }
        }
    }
    //--- den Wert prev_calculated für den nächsten Anruf zurückgeben
    return(rates_total);
}

```

### Sehen Sie auch

[Realtypen \(double, float\)](#), [StringToDouble](#), [DoubleToString](#), [FileWriteDouble](#)

## FileReadFloat

Liest Zahl mit einfacher Genauigkeit mit dem Fließpunkt von der laufenden Position der binären Datei.

```
float FileReadFloat(  
    int file_handle    // Datei-Handle  
);
```

### Parameter

*file\_handle*

[in] Dateiattribut, das durch die Funktion [FileOpen\(\)](#) zurückgegeben wird.

### Rückgabewert

Wert des Typs float.

### Bemerkung

für die Erhaltung der fehlerbezogenen Information muss die Funktion [GetLastError\(\)](#) aufgerufen werden.

**Beispiel** (die Datei wird verwendet, die infolge der Arbeit des Beispiels für die Funktion [FileWriteFloat](#) bekommen ist)

```

//+-----+
//|                                     Demo_FileReadFloat.mq5 |
//|                                     Copyright 2013, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots  1
//---- plot Label1
#property indicator_label1  "CloseLine"
#property indicator_type1   DRAW_COLOR_LINE
#property indicator_color1  clrRed,clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
//--- Parameter zum Lesen von Daten
input string InpFileName="Close.bin"; // der Dateiname
input string InpDirectoryName="Data"; // der Verzeichnisname
//--- globale Variablen
int      ind=0;
int      size=0;
double   close_buff[];
datetime time_buff[];
//--- Indicator Buffers
double   buff[];
double   color_buff[];
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    int def_size=100;
//--- trennen Sie den Speichern für den Arrays
    ArrayResize(close_buff,def_size);
    ArrayResize(time_buff,def_size);
//--- öffnen Sie die Datei
    ResetLastError();
    int file_handle=FileOpen(InpDirectoryName+"\\"+InpFileName,FILE_READ|FILE_BIN);
    if(file_handle!=INVALID_HANDLE)
    {
        PrintFormat("%s Datei ist zum Lesen geöffnet",InpFileName);
        PrintFormat("Pfad zur Datei: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
//--- lesen Sie die Dateidaten
        while(!FileIsEnding(file_handle))
        {
            //--- lesen Sie die Zeitwert und Preiswert
            time_buff[size]=(datetime)FileReadDouble(file_handle);
            close_buff[size]=(double)FileReadFloat(file_handle);
            size++;
            //--- die Größe des Arrays zu erhöhen, wenn sie überfüllt sind
            if(size==def_size)
            {
                def_size+=100;
                ArrayResize(close_buff,def_size);
                ArrayResize(time_buff,def_size);
            }
        }
//--- schließen Sie die Datei
        FileClose(file_handle);
    }
}

```

```

        PrintFormat("Die Daten werden gelesen, %s Datei geschlossen",InpFileName);
    }
    else
    {
        PrintFormat("Fehler beim Öffnen der Datei %s, Fehlercode = %d",InpFileName,GetLastError());
        return (INIT_FAILED);
    }
}

//--- Arrays-Bindung zum Indikator Puffers
SetIndexBuffer(0,buff,INDICATOR_DATA);
SetIndexBuffer(1,color_buff,INDICATOR_COLOR_INDEX);
//---- erstellen Sie die Anzeige-Werte, die im Diagramm nicht sichtbar sein werden
PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
return (INIT_SUCCEEDED);
}

//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
    ArraySetAsSeries(time,false);
    //--- der Zyklus für die noch unbearbeiteten Bars
    for(int i=prev_calculated;i<rates_total;i++)
    {
        //--- die Voreinstellung 0
        buff[i]=0;
        color_buff[i]=0; // die rote Farbe als Voreinstellung
        //--- prüfen Sie, ob es noch Daten gibt
        if(ind<size)
        {
            for(int j=ind;j<size;j++)
            {
                //--- wenn die Daten identisch sind, dann verwenden Sie den Wert aus einer
                if(time[i]==time_buff[j])
                {
                    //--- erhalten Sie einen Preis
                    buff[i]=close_buff[j];
                    //--- wenn der aktuelle Preis größer als die vorherige, so Farbe blau
                    if(buff[i-1]>buff[i])
                        color_buff[i]=1;
                    //--- vergrößern Sie den Zähler
                    ind=j+1;
                    break;
                }
            }
        }
    }
    //--- den Wert prev_calculated für den nächsten Anruf zurückgeben
    return (rates_total);
}

```

Sehen Sie auch

Realtypen (double, float), [FileReadDouble](#), [FileWriteFloat](#)

## FileReadInteger

Liest den Wert des Typs int, short oder char aus der binären Datei abhängig von der angegebenen Länge in Bytes. Das Lesen wird von der laufenden Position des Dateianzeigers durchgeführt.

```
int FileReadInteger(  
    int file_handle,           // Datei-Handle  
    int size=INT_VALUE        // die Größe des ganzzahligen Typs in Bytes  
);
```

### Parameter

*file\_handle*

[in] Dateiattribut, das durch die Funktion [FileOpen\(\)](#) zurückgegeben wird.

*size=INT\_VALUE*

[in] Die Anzahl der Bytes (bis zu 4), die Sie lesen möchten. Die folgenden Konstanten sind verfügbar: CHAR\_VALUE=1, SHORT\_VALUE=2 und INT\_VALUE=4, so kann die Funktion den ganzzahligen Wert des Typs char, short oder int lesen.

### Rückgabewert

Ein Wert vom Typ int. Das Ergebnis dieser Funktion muss deutlich in den Zieltyp, d.h. in den Typ, den Sie lesen möchten, geführt werden. Da der Rückgabewert vom Typ int ist, dann kann er leicht an jede Integer-Wert umgewandelt werden. Der Dateizeiger wird durch die Anzahl der gelesenen Bytes verschoben.

### Bemerkung

Wenn weniger als 4 Bytes gelesen werden, wird das Ergebnis immer positiv sein. Wenn ein oder zwei Bytes gelesen werden, können Sie das Vorzeichen durch eine explizite Umwandlung bzw. in den Typ char (1 Byte) oder short (2 Byte) bestimmen. Bestimmung vom Verzeichnis für die Drei-Byte-Zahl ist nicht trivial, da es keine entsprechenden [Basiswerttyp](#) gibt.

**Beispiel** (die Datei wird verwendet, die infolge der Arbeit des Beispiels für die Funktion [FileWriteInteger](#) bekommen ist)

```
//+-----+
//|                                     Demo_FileReadInteger.mq5 |
//|                                     Copyright 2013, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots  1
//---- plot Label1
#property indicator_label1  "Trends"
#property indicator_type1   DRAW_SECTION
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- Parameter zum Lesen von Daten
input string InpFileName="Trend.bin"; // der Dateiname
input string InpDirectoryName="Data"; // der Verzeichnisname
//--- globale Variablen
int      ind=0;
int      size=0;
datetime time_buff[];
//--- indicator buffers
double   buff[];
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    int def_size=100;
    //--- trennen Sie den Speichern für den Arrays
    ArrayResize(time_buff,def_size);
    //--- öffnen Sie die Datei
    ResetLastError();
    int file_handle=FileOpen(InpDirectoryName+"\\"+InpFileName,FILE_READ|FILE_BIN);
    if(file_handle!=INVALID_HANDLE)
    {
        PrintFormat("%s Datei ist zum Lesen geöffnet",InpFileName);
        PrintFormat("Pfad zur Datei: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
        //--- Hilfsvariablen
        int   arr_size;
        uchar arr[];
        //--- lesen Sie die Dateidaten
        while(!FileIsEnding(file_handle))
        {
            //--- erkennen Sie wieviele Symbole dienen für Zeiterfassung
            arr_size=FileReadInteger(file_handle,INT_VALUE);
            ArrayResize(arr,arr_size);
            for(int i=0;i<arr_size;i++)
                arr[i]=(char)FileReadInteger(file_handle,CHAR_VALUE);
            //--- merken Sie den Zeitwert
            time_buff[size]=StringToTime(CharArrayToString(arr));
            size++;
            //--- die Größe des Arrays zu erhöhen, wenn sie überfüllt sind
            if(size==def_size)
            {
                def_size+=100;
                ArrayResize(time_buff,def_size);
            }
        }
    }
}
```

```

    }
    //--- schließen Sie die Datei
    FileClose(file_handle);
    PrintFormat("Die Daten sind gelesen, die Datei %s geschlossen",InpFileName);
}
else
{
    PrintFormat("Fehler beim Öffnen der Datei %s, Fehlercode = %d",InpFileName,GetLastError());
    return(INIT_FAILED);
}
}
//--- Arrays-Bindung zum Indikator Puffers
SetIndexBuffer(0,buff,INDICATOR_DATA);
//---- erstellen Sie die Indikatorwerte, die im Diagramm nicht sichtbar sein werden
PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
return(INIT_SUCCEEDED);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
    ArraySetAsSeries(time,false);
    ArraySetAsSeries(close,false);
    //--- der Zyklus für die noch unbearbeiteten Bars
    for(int i=prev_calculated;i<rates_total;i++)
    {
        //--- die Voreinstellung 0
        buff[i]=0;
        //--- prüfen Sie, ob es noch Daten gibt
        if(ind<size)
        {
            for(int j=ind;j<size;j++)
            {
                //--- wenn die Daten identisch sind, dann verwenden Sie den Wert aus einer
                if(time[i]==time_buff[j])
                {
                    //--- erhalten Sie einen Preis
                    buff[i]=close[i];
                    //--- vergrößern Sie den Zähler
                    ind=j+1;
                    break;
                }
            }
        }
    }
    //--- den Wert prev_calculated für den nächsten Anruf zurückgeben
    return(rates_total);
}

```

Sehen Sie auch



[IntegerToString](#), [StringToInteger](#), [Ganzzahlige Typen](#), [FileWriteInteger](#)

## FileReadLong

Liest die Ganzzahl des Typs long (8 Byte) von der laufenden Position der binären Datei.

```
long FileReadLong(  
    int file_handle    // Datei-Handle  
);
```

### Parameter

*file\_handle*

[in] Dateiattribut, das durch die Funktion [FileOpen\(\)](#) zurückgegeben wird.

### Rückgabewert

Wert des Typs long.

**Beispiel** (die Datei wird verwendet, die infolge der Arbeit des Beispiels für die Funktion [FileWriteLong](#) bekommen ist)

```
//+-----+
//|                                     Demo_FileReadLong.mq5 |
//|                                     Copyright 2013, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_buffers 1
#property indicator_plots  1
//---- plot Label1
#property indicator_label1  "Volume"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrYellow
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
#property indicator_separate_window
//--- Parameter zum Lesen von Daten
input string InpFileName="Volume.bin"; // der Dateiname
input string InpDirectoryName="Data";  // der Verzeichnisname
//--- globale Variablen
int      ind=0;
int      size=0;
long     volume_buff[];
datetime time_buff[];
//--- indicator buffers
double   buff[];
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- öffnen Sie die Datei
ResetLastError();
int file_handle=FileOpen(InpDirectoryName+"\\"+InpFileName,FILE_READ|FILE_BIN);
if(file_handle!=INVALID_HANDLE)
{
PrintFormat("%s Datei ist zum Lesen geöffnet",InpFileName);
PrintFormat("Pfad zur Datei: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
//--- lesen Sie zuerst wie viele Daten im Datei sind
size=(int)FileReadLong(file_handle);
//--- trennen Sie den Speichern für den Arrays
ArrayResize(volume_buff,size);
ArrayResize(time_buff,size);
//--- lesen Sie den Dateidaten
for(int i=0;i<size;i++)
{
time_buff[i]=(datetime)FileReadLong(file_handle);
volume_buff[i]=FileReadLong(file_handle);
}
//--- schließen Sie die Datei
FileClose(file_handle);
PrintFormat("Die Daten sind gelesen, die Datei %s geschlossen",InpFileName);
}
else
{
PrintFormat("Fehler beim Öffnen der Datei %s, Fehlercode = %d",InpFileName,GetLastError());
return(INIT_FAILED);
}
//--- Array-Bindung zum Indikator Puffer mit 0 Index
SetIndexBuffer(0,buff,INDICATOR_DATA);
```

```

//---- erstellen Sie die Indikatorwerte, die im Diagramm nicht sichtbar sein werden
PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
return(INIT_SUCCEEDED);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
    ArraySetAsSeries(time,false);
    //--- der Zyklus für die noch unbearbeiteten Bars
    for(int i=prev_calculated;i<rates_total;i++)
    {
        //--- Voreinstellung 0
        buff[i]=0;
        //--- prüfen Sie, ob es noch Daten gibt
        if(ind<size)
        {
            for(int j=ind;j<size;j++)
            {
                //--- wenn die Daten identisch sind, dann verwenden Sie den Wert aus einer
                if(time[i]==time_buff[j])
                {
                    buff[i]=(double)volume_buff[j];
                    ind=j+1;
                    break;
                }
            }
        }
    }
    //--- den Wert prev_calculated für den nächsten Anruf zurückgeben
    return(rates_total);
}

```

Sehen Sie auch

[Ganzzahlige Typen](#), [FileReadInteger](#), [FileWriteLong](#))

## FileReadNumber

Liest aus der Datei des Typs CSV die Zeile von der laufenden Position bis zum Begrenzer (oder bis zum Ende der Textzeile) und wandelt die gelesene Zeile in den Wert des Typs double um.

```
double FileReadNumber(  
    int file_handle    // Datei-Handle  
);
```

### Parameter

*file\_handle*

[in] Dateiattribut, das durch die Funktion [FileOpen\(\)](#) zurückgegeben wird.

### Rückgabewert

Wert des Typs double.

**Beispiel** (die Datei wird verwendet, die infolge der Arbeit des Beispiels für die Funktion [FileWriteString](#) bekommen ist)

```

//+-----+
//|                                     Demo_FileReadNumber.mq5 |
//|                                     Copyright 2013, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots 1
//---- plot Label1
#property indicator_label1 "Overbought & Oversold"
#property indicator_type1  DRAW_COLOR_BARS
#property indicator_color1  clrRed, clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
//--- Parameter zum Lesen von Daten
input string InpFileName="RSI.csv"; // der Dateiname
input string InpDirectoryName="Data"; // der Verzeichnisname
//--- die Indikator-Buffers
double   open_buff[];
double   high_buff[];
double   low_buff[];
double   close_buff[];
double   color_buff[];
//--- überkauft Variablen
int       ovb_ind=0;
int       ovb_size=0;
datetime ovb_time[];
//--- überverkauft Variablen
int       ovs_ind=0;
int       ovs_size=0;
datetime ovs_time[];
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- Die standartmäßige Variable Größe von Arrays
    int ovb_def_size=100;
    int ovs_def_size=100;
//--- trennen Sie den Speichern für den Arrays
    ArrayResize(ovb_time,ovb_def_size);
    ArrayResize(ovs_time,ovs_def_size);
//--- öffnen Sie die Datei
    ResetLastError();
    int file_handle=FileOpen(InpDirectoryName+"\\"+InpFileName,FILE_READ|FILE_CSV|FILE_
    if(file_handle!=INVALID_HANDLE)
    {
        PrintFormat(" %s Datei ist zum Lesen geöffnet",InpFileName);
        PrintFormat("Pfad zur Datei: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH)
        double value;
        //--- lesen Sie die Dateidaten
        while(!FileIsEnding(file_handle))
        {
            //--- lesen Sie den ersten Wert in eine Zeichenfolge
            value=FileReadNumber(file_handle);
            //--- lesen Sie die verschiedenen Arrays, abhängig vom Ergebnis der Funktion
            if(value>=70)
                ReadData(file_handle,ovb_time,ovb_size,ovb_def_size);

```

```

        else
            ReadData(file_handle, ovs_time, ovs_size, ovs_def_size);
    }
    //--- schließen Sie die Datei
    FileClose(file_handle);
    PrintFormat("Die Daten werden gelesen, die Datei %s geschlossen", InpFileName);
}
else
{
    PrintFormat("Fehler beim Öffnen der Datei %s, Fehlercode = %d", InpFileName, GetLastError());
    return(INIT_FAILED);
}
}
//--- Arrays-Bindung
SetIndexBuffer(0, open_buff, INDICATOR_DATA);
SetIndexBuffer(1, high_buff, INDICATOR_DATA);
SetIndexBuffer(2, low_buff, INDICATOR_DATA);
SetIndexBuffer(3, close_buff, INDICATOR_DATA);
SetIndexBuffer(4, color_buff, INDICATOR_COLOR_INDEX);
//---- erstellen Sie die Anzeige-Werte, die im Diagramm nicht sichtbar sein werden
PlotIndexSetDouble(0, PLOT_EMPTY_VALUE, 0);
//---
return(INIT_SUCCEEDED);
}
//+-----+
//| Lesen Sie die Zeilen einer Datei |
//+-----+
void ReadData(const int file_handle, datetime &arr[], int &size, int &def_size)
{
    bool flag=false;
    //--- Lesen Sie, bis das Ende der Zeichenfolge oder Datei erreichen
    while(!FileIsLineEnding(file_handle) && !FileIsEnding(file_handle))
    {
        //--- bewegen Sie den Wagen, nach dem Lesen der Anzahl
        if(flag)
            FileReadNumber(file_handle);
        //--- das aktuelle Datum speichern
        arr[size]=FileReadDatetime(file_handle);
        size++;
        //--- Falls erforderlich, vergrößern Sie die Größe des Arrays
        if(size==def_size)
        {
            def_size+=100;
            ArrayResize(arr, def_size);
        }
        //--- übergeben die erste iteration
        flag=true;
    }
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])

```

```
{
```



```

    ArraySetAsSeries(time, false);
    ArraySetAsSeries(open, false);
    ArraySetAsSeries(high, false);
    ArraySetAsSeries(low, false);
    ArraySetAsSeries(close, false);
    ///--- der Zyklus für die noch unbearbeiteten Bars
    for(int i=prev_calculated; i<rates_total; i++)
    {
        ///--- die Voreinstellung 0
        open_buff[i]=0;
        high_buff[i]=0;
        low_buff[i]=0;
        close_buff[i]=0;
        color_buff[i]=0;
        ///--- prüfen Sie, ob es noch Daten gibt
        if(ovb_ind<ovb_size)
            for(int j=ovb_ind; j<ovb_size; j++)
            {
                ///--- wenn die Daten identisch sind, ist die Bar in überkauft Zone
                if(time[i]==ovb_time[j])
                {
                    open_buff[i]=open[i];
                    high_buff[i]=high[i];
                    low_buff[i]=low[i];
                    close_buff[i]=close[i];
                    ///--- 0 - die rote Farbe
                    color_buff[i]=0;
                    ///--- vergrößern Sie den Zähler
                    ovb_ind=j+1;
                    break;
                }
            }
        ///--- prüfen Sie, ob es noch Daten gibt
        if(ovs_ind<ovs_size)
            for(int j=ovs_ind; j<ovs_size; j++)
            {
                ///--- wenn die Daten identisch sind, ist die Bar in überverkauft Zone
                if(time[i]==ovs_time[j])
                {
                    open_buff[i]=open[i];
                    high_buff[i]=high[i];
                    low_buff[i]=low[i];
                    close_buff[i]=close[i];
                    ///--- 1 - die blau Farbe
                    color_buff[i]=1;
                    ///--- vergrößern Sie den Zähler
                    ovs_ind=j+1;
                    break;
                }
            }
    }
    ///--- den Wert prev_calculated für den nächsten Anruf zurückgeben
    return(rates_total);
}
//+-----+
//| Der Bearbeiter des Ereignisses ChartEvent |
//+-----+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam

```

```
        )  
  
    {  
    //--- ändern Sie die Breite des Indikators je nach dem Ausmaß  
    if (ChartGetInteger (0, CHART_SCALE) > 3)  
        PlotIndexSetInteger (0, PLOT_LINE_WIDTH, 2);  
    else  
        PlotIndexSetInteger (0, PLOT_LINE_WIDTH, 1);  
    }
```

Sehen Sie auch

[FileWriteString](#)

## FileReadString

Liest aus der Datei die Zeile von der laufenden Position des Dateianzeigers.

```
string FileReadString(  
    int file_handle,      // Datei-Handle  
    int length=-1        // Zeilenlänge  
);
```

### Parameter

*file\_handle*

[in] Dateiattribut, das durch die Funktion [FileOpen\(\)](#) zurückgegeben wird.

*length=-1*

[in] Anzahl der Symbole für Lesen.

### Rückgabewert

Gelesene Zeile(string).

### Bemerkung

Beim Lesen aus der bin-Datei ist es notwendig, die Länge der gelesenen Zeile anzugeben. Beim Lesen aus der txt-Datei braucht man nicht, die Zeilenlänge zu spezifizieren, die Zeile wird von der laufenden Position bis zum Zeilenvorschubszeichen "\r\n" gelesen". Beim Lesen aus der csv-Datei braucht man auch nicht die Zeilenlänge anzugeben, die Zeile wird von der laufenden Position bis zum nächsten Begrenzer oder bis zum Endezeichen der Textzeile.

Wenn die Datei mit [Flagge](#) FILE\_ANSI geöffnet wird, wird die gelesene Zeile in Unicode umgewandelt.

**Beispiel** (die Datei wird verwendet, die infolge der Arbeit des Beispiels für die Funktion [FileWriteInteger](#) bekommen ist)

```

//--- Beim Starten des Scripts zeigen Sie den Fenster mit den Eingabeparametern an
#property script_show_inputs
//--- Parameter zum Lesen von Daten
input string InpFileName="Trend.bin"; // der Dateiname
input string InpDirectoryName="Data"; // der Verzeichnisname
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//--- öffnen Sie die Datei
ResetLastError();
int file_handle=FileOpen(InpDirectoryName+"\\"+InpFileName,FILE_READ|FILE_BIN|FILE_
if(file_handle!=INVALID_HANDLE)
{
PrintFormat("%s Datei ist zum Lesen geöffnet",InpFileName);
PrintFormat("Pfad zur Datei: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH)
//--- Hilfsvariablen
int str_size;
string str;
//--- lesen Sie die Dateidaten
while(!FileIsEnding(file_handle))
{
//--- erkennen Sie wieviele Symbole dienen für Zeiterfassung
str_size=FileReadInteger(file_handle,INT_VALUE);
//--- lesen Sie die Zeile
str=FileReadString(file_handle,str_size);
//--- drucken Sie die Zeile
PrintFormat(str);
}
//--- schließen Sie die Datei
FileClose(file_handle);
PrintFormat("Die Daten sind gelesen, die Datei %s geschlossen",InpFileName);
}
else
PrintFormat("Fehler beim Öffnen der Datei %s, Fehlercode = %d",InpFileName,GetL
}

```

Sehen Sie auch

[Typ string](#), [Datenverarbeitung](#), [FileWriteInteger](#)

## FileReadStruct

Liest den Inhalt aus der binären Datei in die Struktur, die als Parameter übertragen wird. Lesen wird von der laufenden Position des Dateianzeigers durchgeführt.

```
uint FileReadStruct(  
    int          file_handle,          // Datei-Handle  
    const void&  struct_object,       // Zielstruktur für Einlesen  
    int          size=-1               // Strukturgröße in Bytes  
);
```

### Parameter

*file\_handle*

[in] Dateittribut der geöffneten binären Datei .

*struct\_object*

[out] Objekt der angegebenen Struktur. Struktur muss keine Zeilen, [dynamische Felder](#) und [virtuelle Funktionen](#) enthalten.

*size=-1*

[in] Anzahl der Bytes, die eingelesen werden müssen. Wenn die Größe nicht angegeben wird oder die Anzahl der angegebenen Bytes größer als die Strukturgröße ist, wird die genaue Größe der angegebenen Struktur verwendet.

### Rückgabewert

Im Erfolgsfall gibt die Funktion die Anzahl der eingelesenen Bytes zurück. Dateianzeiger wird um dieselbe Zahl der Bytes verschoben.

**Beispiel** (die Datei wird verwendet, die infolge der Arbeit des Beispiels für die Funktion [FileWriteStruct](#) bekommen ist)

```
//+-----+
//|                                     Demo_FileReadStruct.mq5 |
//|                                     Copyright 2013, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots  1
//---- plot Label1
#property indicator_label1  "Candles"
#property indicator_type1   DRAW_CANDLES
#property indicator_color1  clrOrange
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
#property indicator_separate_window
//--- Die Parameter für das Erhalten der Daten
input string  InpFileName="EURUSD.txt"; // der Dateiname
input string  InpDirectoryName="Data";  // der Verzeichnisname
//+-----+
//| Struktur für die Speicherung von Daten-Kerzen |
//+-----+
struct candlesticks
{
    double      open; // Eröffnungspreis
    double      close; // Schlußpreis
    double      high; // der maximale Preis
    double      low; // der minimale Preis
    datetime    date; // Datum
};
//--- Indikator Buffers
double      open_buff[];
double      close_buff[];
double      high_buff[];
double      low_buff[];
//--- globale Variablen
candlesticks cand_buff[];
int          size=0;
int          ind=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    int default_size=100;
    ArrayResize(cand_buff,default_size);
//--- öffnen Sie die Datei
ResetLastError();
int file_handle=FileOpen(InpDirectoryName+"\\"+InpFileName,FILE_READ|FILE_BIN|FILE_
if(file_handle!=INVALID_HANDLE)
{
    PrintFormat("%s Datei ist zum Lesen geöffnet",InpFileName);
    PrintFormat("Pfad zur Datei: %s\\Files\\",TerminalInfoString(TERMINAL_COMMONDATA
//--- lesen Sie die Dateidaten
while(!FileIsEnding(file_handle))
{
    //--- zeichnen Sie die Dateien in den Array auf
    FileReadStruct(file_handle,cand_buff[size]);
    size++;
}
```

```

    //--- prüfen Sie das Array für Überbelegung
    if(size==default_size)
    {
        //--- erhöhen die Dimensionalität des Arrays
        default_size+=100;
        ArrayResize(cand_buff,default_size);
    }
}
//--- schließen Sie die Datei
FileClose(file_handle);
PrintFormat("Die Daten sind gelesen, die Datei %s geschlossen",InpFileName);
}
else
{
    PrintFormat("Fehler beim Öffnen der Datei %s, Fehlercode = %d",InpFileName,GetLastError());
    return(INIT_FAILED);
}
//--- Bindung von Array zum Indikator-Puffer
SetIndexBuffer(0,open_buff,INDICATOR_DATA);
SetIndexBuffer(1,high_buff,INDICATOR_DATA);
SetIndexBuffer(2,low_buff,INDICATOR_DATA);
SetIndexBuffer(3,close_buff,INDICATOR_DATA);
//--- der Leerwert
PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
return(INIT_SUCCEEDED);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{

```

```
    ArraySetAsSeries(time, false);  
    ///--- der Zyklus für die noch unbearbeiteten Kerzen  
    for(int i=prev_calculated; i<rates_total; i++)  
    {  
        ///--- die Voreinstellung 0  
        open_buff[i]=0;  
        close_buff[i]=0;  
        high_buff[i]=0;  
        low_buff[i]=0;  
        ///--- prüfen Sie, ob es noch Daten gibt  
        if(ind<size)  
        {  
            for(int j=ind; j<size; j++)  
            {  
                ///--- wenn die Daten identisch sind, dann verwenden Sie den Wert aus einer  
                if(time[i]==cand_buff[j].date)  
                {  
                    open_buff[i]=cand_buff[j].open;  
                    close_buff[i]=cand_buff[j].close;  
                    high_buff[i]=cand_buff[j].high;  
                    low_buff[i]=cand_buff[j].low;  
                    ///--- vergrößern Sie den Zähler  
                    ind=j+1;  
                    break;  
                }  
            }  
        }  
    }  
    ///--- den Wert prev_calculated für den nächsten Anruf zurückgeben  
    return(rates_total);  
}
```

Sehen Sie auch

[Strukturen und Klassen](#), [FileWriteStruct](#)



## FileSeek

Verschiebt die Position des Dateianzeigers um die angegebene Anzahl der Bytes gegen die angegebene Position.

```
bool FileSeek(  
    int          file_handle,    // Datei-Handle  
    long         offset,        // in Bytes  
    ENUM_FILE_POSITION origin    // Position für Abzählung  
);
```

### Parameter

*file\_handle*

[in] Dateiattribut, das durch die Funktion [FileOpen\(\)](#) zurückgegeben wird.

*offset*

[in] Übertragung in Bytes (kann auch negativen Wert annehmen).

*origin*

[in] Anfangspunkt für Übertragung. Kann einen der Aufzählungswerte [ENUM\\_FILE\\_POSITION](#) annehmen.

### Rückgabewert

Gibt true im Erfolgsfall zurück, anderenfalls false. Für die Erhaltung der [fehlerbezogenen](#) Information muss die Funktion [GetLastError\(\)](#) aufgerufen werden.

### Bemerkung

Wenn das Ergebnis der Funktionsdurchführung FileSeek() eine negative Zahl ist (außerhalb der "linken Grenze" der Datei), wird Dateianzeiger am Dateianfang gestellt.

Wenn die Position außerhalb der "rechten Grenze" der Datei (mehr als Dateigröße) gestellt wird, wird das nächste Schreiben in die Datei nicht von der Dateiende, sondern von der gestellten Position durchgeführt. Dabei werden unbestimmte Werte zwischen dem früheren Dateiende und der gestellten Position geschrieben.

### Beispiel:

```
//+-----+
//|                                     Demo_FileSeek.mq5 |
//|                                     Copyright 2013, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- Beim Starten des Scripts zeigen Sie den Fenster mit den Eingabeparametern an
#property script_show_inputs
//--- Eingabeparameters
input string InpFileName="file.txt";    // der Dateiname
input string InpDirectoryName="Data";   // der Verzeichnisname
input int    InpEncodingType=FILE_ANSI; // ANSI=32 oder UNICODE=64
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//--- feststellen Sie den Variablenwert um Zufallszahlen zu generieren
    _RandomSeed=GetTickCount();
//--- Variabel für die Positionen des Anfanges der Zeilen
    ulong pos[];
    int size;
//--- stürzen Sie den Fehlerwert
    ResetLastError();
//--- öffnen Sie die Datei
    int file_handle=FileOpen(InpDirectoryName+"\\"+InpFileName,FILE_READ|FILE_TXT|InpEn
    if(file_handle!=INVALID_HANDLE)
    {
        PrintFormat("%s Datei ist zum Lesen geöffnet",InpFileName);
        //--- bekommen Sie die Position des Anfanges für jede Zeile in der Datei
        GetStringPositions(file_handle,pos);
        //--- ermitteln Sie wieviel allen der Zeilen in der Datei sind
        size=ArraySize(pos);
        if(!size)
        {
            //--- Wenn es in der Datei keine Zeilen gibt, so beenden wir die Arbeit
            PrintFormat("Die Datei %s ist leer!",InpFileName);
            FileClose(file_handle);
            return;
        }
        //--- wählen Sie zufällig die Nummer der Zeile
        int ind=MathRand()%size;
        //--- verschieben Sie die Position an den Anfang der Zeile
        if(FileSeek(file_handle,pos[ind],SEEK_SET)==true)
        {
            //--- lesen und drucken Sie eine Zeile mit der Nummer ind
            PrintFormat("Der Text der Zeile mit der Nummer %d: \"%s\"",ind,FileReadString
        }
        //--- schließen Sie die Datei
        FileClose(file_handle);
        PrintFormat("die Datei %s ist geschlossen",InpFileName);
    }
    else
        PrintFormat("Fehler beim Öffnen der Datei %s, Fehlercode = %d",InpFileName,GetLe
}
//+-----+
//| Funktion bestimmt die Startposition für jede der Zeilen in der |
```

```
///| Datei und legt sie in das Array arr |
///+-----+
void GetStringPositions(const int handle,ulong &arr[])
{
    ///--- die standardmässige Größe des Arrays
    int def_size=127;
    ///--- trennen Sie den Speichern für den Arrays
    ArrayResize(arr,def_size);
    ///--- Der Zähler der Zeilen
    int i=0;
    ///--- Wenn nicht das Ende der Datei, dann es gibt mindestens eine Zeile
    if(!FileIsEnding(handle))
    {
        arr[i]=FileTell(handle);
        i++;
    }
    else
        return; // die Datei ist leer, gehen
    ///--- bestimmen Sie die Verschiebung in den Bytes je nach der Kodierung
    int shift;
    if(FileGetInteger(handle,FILE_IS_ANSI))
        shift=1;
    else
        shift=2;
    ///--- im Zyklus lesen Sie die Zeilen aus
    while(1)
    {
        ///--- lesen Sie die Zeile
        FileReadString(handle);
        ///--- Die Prüfung auf dem Dateiende
        if(!FileIsEnding(handle))
        {
            ///--- merken Sie die Position der nächsten Zeile
            arr[i]=FileTell(handle)+shift;
            i++;
            ///--- die Größe des Arrays zu erhöhen, wenn es überfüllt ist
            if(i==def_size)
            {
                def_size+=def_size+1;
                ArrayResize(arr,def_size);
            }
        }
        else
            break; // der Dateiende, gehen
    }
    ///--- feststellen Sie den wahrhaften Umfang des Arrays
    ArrayResize(arr,i);
}
```

## FileSize

Gibt die Größe der Datei in Bytes zurück.

```
ulong FileSize(  
    int file_handle    // Datei-Handle  
);
```

### Parameter

*file\_handle*

[in] Dateiattribut, das durch die Funktion [FileOpen\(\)](#) zurückgegeben wird.

### Rückgabewert

Wert des Typs int.

### Bemerkung

Für die Erhaltung der [fehlerbezogenen](#) Information muss die Funktion [GetLastError\(\)](#) aufgerufen werden.

### Beispiel:

```

//--- Beim Starten des Scripts zeigen Sie den Fenster mit den Eingabeparametern an
#property script_show_inputs
//--- Eingabeparameters
input ulong   InpThresholdSize=20;           // die Grenze von Dateigröße im Kilobyte
input string  InpBigFolderName="big";        // Ordner für die größten Dateien
input string  InpSmallFolderName="small";    // Ordner für die kleinen Dateien
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    string   file_name;           // Variable für Speichern die Dateinamen
    string   filter="*.csv";      // Der Filter für die Suche nach Dateien
    ulong    file_size=0;         // die Größe der Datei in Bytes
    int      size=0;              // die Anzahl der Dateien
    //--- drucken Sie den Weg zum Ordner in der arbeiten werden
    PrintFormat("Arbeiten in dem Ordner %s\\Files\\",TerminalInfoString(TERMINAL_COMMON));
    //--- erhalten Sie die Suche Handle von Grund auf den Gesamtordner aller Kundentermine
    long search_handle=FileFindFirst(filter,file_name,FILE_COMMON);
    //--- prüfen Sie, ob die FileFindFirst() Funktion erfolgreich durchgeführt hat
    if(search_handle!=INVALID_HANDLE)
    {
        //--- im Zyklus versetzen Sie die Dateien je nach ihrer Größe
        do
        {
            //--- öffnen Sie die Datei
            ResetLastError();
            int file_handle=FileOpen(file_name,FILE_READ|FILE_CSV|FILE_COMMON);
            if(file_handle!=INVALID_HANDLE)
            {
                //--- erhalten Sie die Größe der Datei
                file_size=FileSize(file_handle);
                //--- schließen Sie die Datei
                FileClose(file_handle);
            }
            else
            {
                PrintFormat("Fehler beim Öffnen der Datei %s, Fehlercode = %d",file_name,GetLastError());
                continue;
            }
            //--- drucken Sie die Größe der Datei
            PrintFormat("die Größe der Datei %s ist %d Byte gleich",file_name,file_size);
            //--- ermitteln Sie den Pfad um die Datei zu versetzen
            string path;
            if(file_size>InpThresholdSize*1024)
                path=InpBigFolderName+"\\"+file_name;
            else
                path=InpSmallFolderName+"\\"+file_name;
            //--- versetzen Sie die Datei
            ResetLastError();
            if(FileMove(file_name,FILE_COMMON,path,FILE_REWRITE|FILE_COMMON))
                PrintFormat("%s Datei ist versetzt",file_name);
            else
                PrintFormat("Fehler, Kode = %d",GetLastError());
        }
        while(FileFindNext(search_handle,file_name));
        //--- schließen Sie die Suche Handle
        FileFindClose(search_handle);
    }
    else
        Print("Dateien sind nicht geöffnet!");
}

```

```
}
```

## FileTell

Gibt die laufende Position des Dateianzeigers der entsprechenden geöffneten Datei.

```
ulong FileTell(  
    int file_handle    // Datei-Handle  
);
```

### Parameter

*file\_handle*

[in] Dateiattribut, das durch die Funktion [FileOpen\(\)](#) zurückgegeben wird.

### Rückgabewert

Laufende Position des Dateiattributs in Bytes vom Anfang der Datei.

### Bemerkung

Für die Erhaltung der [fehlerbezogenen](#) Information muss die Funktion [GetLastError\(\)](#) aufgerufen werden.

### Beispiel:

```
//+-----+
//|                                     Demo_FileTell.mq5 |
//|                                     Copyright 2013, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- Beim Starten des Scripts zeigen Sie den Fenster mit den Eingabeparametern an
#property script_show_inputs
//--- Eingabeparameters
input string InpFileName="file.txt";    // der Dateiname
input string InpDirectoryName="Data";    // der Verzeichnisname
input int    InpEncodingType=FILE_ANSI; // ANSI=32 oder UNICODE=64
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//--- feststellen Sie den Variablenwert um Zufallszahlen zu generieren
    _RandomSeed=GetTickCount();
//--- Variabel für die Positionen des Anfanges der Zeilen
    ulong pos[];
    int size;
//--- stürzen Sie den Fehlerwert
    ResetLastError();
//--- öffnen Sie die Datei
    int file_handle=FileOpen(InpDirectoryName+"\\"+InpFileName,FILE_READ|FILE_TXT|InpEn
    if(file_handle!=INVALID_HANDLE)
    {
        PrintFormat("%s Datei ist zum Lesen geöffnet",InpFileName);
        //--- bekommen Sie die Position des Anfanges für jede Zeile in der Datei
        GetStringPositions(file_handle,pos);
        //--- ermitteln Sie wieviel allen der Zeilen in der Datei sind
        size=ArraySize(pos);
        if(!size)
        {
            //--- Wenn es in der Datei keine Zeilen gibt, so beenden wir die Arbeit
            PrintFormat("Die Datei %s ist leer!",InpFileName);
            FileClose(file_handle);
            return;
        }
        //--- wählen Sie zufällig die Nummer der Zeile
        int ind=MathRand()%size;
        //--- verschieben Sie die Position an den Anfang der Zeile
        FileSeek(file_handle,pos[ind],SEEK_SET);
        //--- lesen und drucken Sie eine Zeile mit der Nummer ind
        PrintFormat("Der Text der Zeile mit der Nummer %d: \"%s\"",ind,FileReadString(fi
        //--- schließen Sie die Datei
        FileClose(file_handle);
        PrintFormat("die Datei %s ist geschlossen",InpFileName);
    }
    else
        PrintFormat("Fehler beim Öffnen der Datei %s, Fehlercode = %d",InpFileName,GetL
}
//+-----+
//| Funktion bestimmt die Startposition für jede der Zeilen in der |
```



```

//| Datei und legt sie in das Array arr |
//+-----+
void GetStringPositions(const int handle,ulong &arr[])
{
//--- die standardmässige Größe des Arrays
    int def_size=127;
//--- trennen Sie den Speichern für den Arrays
    ArrayResize(arr,def_size);
//--- Der Zähler der Zeilen
    int i=0;
//--- Wenn nicht das Ende der Datei, dann es gibt mindestens eine Zeile
    if(!FileIsEnding(handle))
    {
        arr[i]=FileTell(handle);
        i++;
    }
    else
        return; // die Datei ist leer, gehen
//--- bestimmen Sie die Verschiebung in den Bytes je nach der Kodierung
    int shift;
    if(FileGetInteger(handle,FILE_IS_ANSI))
        shift=1;
    else
        shift=2;
//--- im Zyklus lesen Sie die Zeilen aus
    while(1)
    {
        //--- lesen Sie die Zeile
        FileReadString(handle);
        //--- Die Prüfung auf dem Dateieinde
        if(!FileIsEnding(handle))
        {
            //--- merken Sie die Position der nächsten Zeile
            arr[i]=FileTell(handle)+shift;
            i++;
            //--- die Größe des Arrays zu erhöhen, wenn es überfüllt ist
            if(i==def_size)
            {
                def_size+=def_size+1;
                ArrayResize(arr,def_size);
            }
        }
        else
            break; // der Dateieinde, gehen
    }
//--- feststellen Sie den wahrhaften Umfang des Massives
    ArrayResize(arr,i);
}

```

## FileWrite

Aufzeichnung der Daten in die Datei des Typs CSV oder TXT, der Datenbegrenzer wird automatisch eingefügt, wenn er nicht 0 gleich ist. Nach Aufzeichnung in die Datei wird Zeilenendesymbol "\r\n" hinzugefügt.

```
uint FileWrite(  
    int file_handle,      // Datei-Handle  
    ...                  // Liste der geschriebenen Parameter  
);
```

### Parameter

*file\_handle*

[in] Dateidescriptor, der durch die Funktion [FileOpen\(\)](#) zurückgegeben wird.

...

[in] Liste der Parameter, die durch Kommas getrennt werden, für Schreiben in die Datei. Anzahl der in die Datei geschriebenen Parameter muss nicht mehr als 63 sein.

### Rückgabewert

Anzahl der geschriebenen Bytes.

### Bemerkung

Bei der Ausgabe werden numerische Daten in Textformat umgewandelt (s. die Funktion [Print\(\)](#)). Daten des Typs double werden mit der Genauigkeit bis 16 Dezimalzeichen ausgegeben, dabei können Daten entweder im traditionellen oder im wissenschaftlichen Format ausgegeben - abhängig davon, welche Aufzeichnung kompakter ist. Daten des Typs werden mit 5 Dezimalzeichen ausgegeben. Für Ausgabe der Realzahlen mit anderer Genauigkeit oder im explizit angegebenen Format muss die Funktion [DoubleToString\(\)](#) verwendet werden.

Zahlen des Typs bool werden als die Zeilen "true" oder "false" ausgegeben. Zahlen des Typs datetime werden im Format "YYYY.MM.DD HH:MI:SS" ausgegeben.

### Beispiel:

```

//+-----+
//|                                     Demo_FileWrite.mq5 |
//|                                     Copyright 2013, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- Beim Starten des Scripts zeigen Sie den Fenster mit den Eingabeparametern an
#property script_show_inputs
//--- die Parameter für das Erhalten der Daten aus dem Terminal
input string      InpSymbolName="EURUSD";           // Währungspaar
input ENUM_TIMEFRAMES InpSymbolPeriod=PERIOD_H1;    // Zeitspanne
input int          InpFastEMAPeriod=12;             // die Periode des schnellen
input int          InpSlowEMAPeriod=26;            // die Periode des langsamen
input int          InpSignalPeriod=9;              // die durchschnittliche Periode
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE; // der Typ des Preises
input datetime     InpDateStart=D'2012.01.01 00:00'; // das Anfangsdatum zum Kopieren
//--- die Parameter für die Aufzeichnung der Daten in die Datei
input string       InpFileName="MACD.csv"; // der Dateiname
input string       InpDirectoryName="Data"; // der Verzeichnisname
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    datetime date_finish; // das Enddatum zum Kopieren von Daten
    bool      sign_buff[]; // ein Array von Signalen (true - kaufen, false - verkaufen)
    datetime  time_buff[]; // ein Array der Zeit des Eintritts der Signale
    int       sign_size=0; // die Größe der Array von Signalen
    double    macd_buff[]; // ein Array von Werten des Indikators
    datetime  date_buff[]; // ein Array von Daten des Indikators
    int       macd_size=0; // die Größe der Array von Indikator
    //--- Die Zeit des Abschlusses - laufend
    date_finish=TimeCurrent();
    //--- bekommen Sie die Handle des Indikators MACD
    ResetLastError();
    int macd_handle=iMACD(InpSymbolName,InpSymbolPeriod,InpFastEMAPeriod,InpSlowEMAPeriod);
    if(macd_handle==INVALID_HANDLE)
    {
        //--- Es misslang, die Handle des Indikators zu bekommen
        PrintFormat("Fehler beim Erhalten der Handle des Indikators. Fehlercode = %d",GetLastError());
        return;
    }
    //--- befinden Sie sich im Zyklus, bis der Indikator alle Werten rechnen wird
    while(BarsCalculated(macd_handle)==-1)
        Sleep(10); // die Verzögerungszeit, damit der Indikator dazugekommen ist die Werte
    //--- Kopieren Sie die Werte eines Indikators für eine bestimmte Periode
    ResetLastError();
    if(CopyBuffer(macd_handle,0,InpDateStart,date_finish,macd_buff)==-1)
    {
        PrintFormat("Fehler beim Kopieren der Werte des Indikators. Fehlercode = %d",GetLastError());
        return;
    }
    //--- kopieren Sie die entsprechende Zeit für die Werte des Indikators
    ResetLastError();
    if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,date_buff)==-1)
    {
        PrintFormat("Fehler beim Kopieren von Zeitwerten. Fehlercode = %d",GetLastError());
        return;
    }
}

```

```

//--- befreien Sie das Gedächtnis, das mit dem Indikator eingenommen wird
IndicatorRelease(macd_handle);
//--- erhalten Sie die Größe des Buffers
macd_size=ArraySize(macd_buff);
//--- Analysieren der Daten und speichern die Indikator-Signale in Arrays
ArrayResize(sign_buff,macd_size-1);
ArrayResize(time_buff,macd_size-1);
for(int i=1;i<macd_size;i++)
{
    //--- Das Signal auf den Kauf
    if(macd_buff[i-1]<0 && macd_buff[i]>=0)
    {
        sign_buff[sign_size]=true;
        time_buff[sign_size]=date_buff[i];
        sign_size++;
    }
    //--- Das Signal auf den Verkauf
    if(macd_buff[i-1]>0 && macd_buff[i]<=0)
    {
        sign_buff[sign_size]=false;
        time_buff[sign_size]=date_buff[i];
        sign_size++;
    }
}
//--- öffnen Sie die Datei zum Schreiben die Indikatorwerte (wenn nicht, dann sie auto
ResetLastError();
int file_handle=FileOpen(InpDirectoryName+"//" +InpFileName,FILE_READ|FILE_WRITE|FILE_APPEND);
if(file_handle!=INVALID_HANDLE)
{
    PrintFormat("%s Datei ist zum Lesen geöffnet",InpFileName);
    PrintFormat("Pfad zur Datei: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
    //--- schreiben Sie zuerst die Anzahl der Signale
    FileWrite(file_handle,sign_size);
    //--- zeichnen Sie die Zeit der Signale und ihre Werte in der Datei auf
    for(int i=0;i<sign_size;i++)
        FileWrite(file_handle,time_buff[i],sign_buff[i]);
    //--- schließen Sie die Datei
    FileClose(file_handle);
    PrintFormat("Die Daten sind aufgezeichnet, die Datei %s geschlossen",InpFileName);
}
else
    PrintFormat("Fehler beim Öffnen der Datei %s, Fehlercode = %d",InpFileName,GetLastError());
}

```

Sehen Sie auch

[Comment](#), [Print](#), [StringFormat](#)

## FileWriteArray

Schreibt in die Datei des Typs BIN Arrays verschiedener Typen ausser Zeilentypen (kann Feld der Strukturen sein, das keine Zeilen und dynamische Arrays enthält).

```
uint FileWriteArray(  
    int          file_handle,          // Datei-Handle  
    const void&  array[],              // Feld  
    int          start=0,              // Anfangsindex im Feld  
    int          count=WHOLE_ARRAY    // Anzahl der Elemente  
);
```

### Parameter

*file\_handle*

[in] Dateiattribut, das durch die Funktion [FileOpen\(\)](#) zurückgegeben wird.

*array[]*

[out] Feld für Aufzeichnung.

*start=0*

[in] Anfangsindex im Feld (Nummer des ersten aufgezeichneten Elementes).

*count=WHOLE\_ARRAY*

[in] Anzahl der zu aufgezeichneten Elemente ([WHOLE\\_ARRAY](#) bedeutet, dass alle Elemente seit der Nummer start bis zum Arraysende aufgezeichnet werden).

### Rückgabewert

Anzahl der aufgezeichneten Elemente.

### Bemerkung

Zeilenfeld kann nur in die Datei des Typs TXT geschrieben werden. In diesem Fall werden die Zeilen automatisch mit Symbolen der Zeilenende "\r\n" beendet. Abhängig vom Dateityp ANSI oder UNICODE, werden die Zeilen zur ansi-Kodierung reduziert oder nicht.

### Beispiel:

```

//+-----+
//|                                     Demo_FileWriteArray.mq5 |
//|                                     Copyright 2013, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- Eingabeparameters
input string InpFileName="data.bin";
input string InpDirectoryName="SomeFolder";
//+-----+
//| Struktur für die Speicherung von Daten auf die Preise |
//+-----+
struct prices
{
    datetime    date; // Datum
    double      bid;  // der Bid Preis
    double      ask;  // der Ask Preis
};
//--- globale Variablen
int    count=0;
int    size=20;
string path=InpDirectoryName+"\\"+InpFileName;
prices arr[];
//+-----+
//| Expert initialization function |
//+-----+
int OnInit()
{
    //--- trennen Sie den Speichern für den Arrays
    ArrayResize(arr,size);
    //---
    return(INIT_SUCCEEDED);
}
//+-----+
//| Expert deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
    //--- Aufzeichnung die verbleibenden count Zeilen, wenn count<n
    WriteData(count);
}
//+-----+
//| Expert tick function |
//+-----+
void OnTick()
{
    //--- Speichern Sie die Daten in einem Array
    arr[count].date=TimeCurrent();
    arr[count].bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
    arr[count].ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
    //--- zeigen Sie aktuelle Daten an
    Print("Date = ",arr[count].date," Bid = ",arr[count].bid," Ask = ",arr[count].ask);
    //--- vergrößern Sie den Zähler
    count++;
    //--- Wenn das Array ausgefüllt wurde, dann die Daten in einer Datei speichern und neu
    if(count==size)
    {
        WriteData(size);
        count=0;
    }
}

```

```
    }  
  }  
  //+-----+  
  //| Aufzeichnung die n Elementen des Arrays in der Datei |  
  //+-----+  
  void WriteData(const int n)  
  {  
    //--- öffnen Sie die Datei  
    ResetLastError();  
    int handle=FileOpen(path,FILE_READ|FILE_WRITE|FILE_BIN);  
    if(handle!=INVALID_HANDLE)  
    {  
      //--- schreiben Sie des Arrays Daten an das Ende der Datei  
      FileSeek(handle,0,SEEK_END);  
      FileWriteArray(handle,arr,0,n);  
      //--- schließen Sie die Datei  
      FileClose(handle);  
    }  
    else  
      Print("Fehler beim Öffnen der Datei, Fehler ",GetLastError());  
  }  
}
```

Sehen Sie auch

[Variablen](#), [FileSeek](#)

## FileWriteDouble

Schreibt in Datei den Wert des Parameters des Typs double von der laufenden Position des Dateianzeigers.

```
uint FileWriteDouble(  
    int     file_handle,    // Datei-Handle  
    double  value           // Werte für Aufzeichnung  
);
```

### Parameter

*file\_handle*

[in] Dateiattribut, das durch die Funktion [FileOpen\(\)](#) zurückgegeben wird.

*value*

[in] Wert des Typs double.

### Rückgabewert

Im Erfolgsfall gibt die Funktion die Anzahl der geschriebenen Bytes zurück (in diesem Fall [sizeof\(double\)](#)=8). Dateianzeiger wird um dieselbe Anzahl der Bytes verschoben.

**Beispiel:**



```
//+-----+
//|                                     Demo_FileWriteDouble.mq5 |
//|                                     Copyright 2013, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- Beim Starten des Scripts zeigen Sie den Fenster mit den Eingabeparametern an
#property script_show_inputs
//--- die Parameter für das Erhalten der Daten aus dem Terminal
input string      InpSymbolName="EURJPY";           // Währungspaar
input ENUM_TIMEFRAMES InpSymbolPeriod=PERIOD_M15;   // Zeitspanne
input int          InpMAPeriod=10;                   // die Glätten Periode
input int          InpMASHift=0;                     // die Absetzung des Indikators
input ENUM_MA_METHOD InpMAMethod=MODE_SMA;          // der Glätten Typ
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE; // der Typ des Preises
input datetime     InpDateStart=D'2013.01.01 00:00'; // das Anfangsdatum zum Kopieren
//--- die Parameter für die Aufzeichnung der Daten in die Datei
input string       InpFileName="MA.csv";             // der Dateiname
input string       InpDirectoryName="Data";         // der Verzeichnisname
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    datetime date_finish=TimeCurrent();
    double    ma_buff[];
    datetime  time_buff[];
    int       size;
//--- bekommen Sie die Handle des Indikators MA
    ResetLastError();
    int ma_handle=iMA(InpSymbolName,InpSymbolPeriod,InpMAPeriod,InpMASHift,InpMAMethod,
    if(ma_handle==INVALID_HANDLE)
    {
        //--- Es misslang, die Handle des Indikators zu bekommen
        PrintFormat("Fehler beim Erhalten der Handle des Indikators. Fehlercode = %d",GetLastError());
        return;
    }
//--- befinden Sie sich im Zyklus, bis der Indikator alle Werten rechnen wird
    while(BarsCalculated(ma_handle)==-1)
        Sleep(20); // die Verzögerungszeit, damit der Indikator dazugekommen ist die Werte
    PrintFormat("die Werte des Indikators, da seit %s beginnen, werden in die Datei aufgeschrieben");
//--- Kopieren Sie die Werte eines Indikators
    ResetLastError();
    if(CopyBuffer(ma_handle,0,InpDateStart,date_finish,ma_buff)==-1)
    {
        PrintFormat("Fehler beim Kopieren der Werte des Indikators. Fehlercode = %d",GetLastError());
        return;
    }
//--- Kopieren Sie die Zeit des Erscheinens der entsprechenden Bars
    ResetLastError();
    if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
    {
        PrintFormat("Fehler beim Kopieren von Zeitwerte. Fehlercode = %d",GetLastError());
        return;
    }
//--- erhalten Sie die Größe des Buffers
    size=ArraySize(ma_buff);
//--- befreien Sie das Gedächtnis, das mit dem Indikator eingenommen wird
    IndicatorRelease(ma_handle);
}
```

```

//--- öffnen Sie die Datei zum Schreiben die Indikatorwerte (wenn nicht, dann sie auto
ResetLastError();
int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
if(file_handle!=INVALID_HANDLE)
{
    PrintFormat("%s Datei ist zum Lesen geöffnet",InpFileName);
    PrintFormat("Pfad zur Datei: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH)
    //--- schreiben Sie zuerst die Größe der Beispieldaten
    FileWriteDouble(file_handle,(double)size);
    //--- schreiben Sie die Zeit und die Indikatorwerte in der Datei
    for(int i=0;i<size;i++)
    {
        FileWriteDouble(file_handle,(double)time_buff[i]);
        FileWriteDouble(file_handle,ma_buff[i]);
    }
    //--- schließen Sie die Datei
    FileClose(file_handle);
    PrintFormat("Die Daten sind aufgezeichnet, die Datei %s geschlossen",InpFileName
}
else
    PrintFormat("Fehler beim Öffnen der Datei %s, Fehlercode = %d",InpFileName,GetLe
}

```

Sehen Sie auch

[Realtypen \(double, float\)](#)

## FileWriteFloat

Schreibt den Parameterwert des Typs float in bin-Datei von der laufenden Position des Dateianzeigers.

```
uint FileWriteFloat(  
    int    file_handle,    // Datei-Handle  
    float  value           // geschriebener Wert  
);
```

### Parameter

*file\_handle*

[in] Dateiattribut, das durch die Funktion [FileOpen\(\)](#) zurückgegeben wird.

*value*

[in] Wert des Typs float.

### Rückgabewert

Im Erfolgsfall gibt die Funktion die Anzahl der geschriebenen Bytes (in diesem Fall [sizeof\(float\)](#)=4).  
Dateianzeiger wird um dieselbe Zahl der Bytes verschiebt.

### Beispiel:

```

//+-----+
//|                                     Demo_FileWriteFloat.mq5 |
//|                                     Copyright 2013, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- Beim Starten des Scripts zeigen Sie den Fenster mit den Eingabeparametern an
#property script_show_inputs
//--- die Parameter für das Erhalten der Daten aus dem Terminal
input string      InpSymbolName="EURUSD";           // Währungspaar
input ENUM_TIMEFRAMES InpSymbolPeriod=PERIOD_M15;    // Zeitspanne
input datetime     InpDateStart=D'2013.01.01 00:00'; // das Anfangsdatum zum Kopieren
//--- die Parameter für die Aufzeichnung der Daten in die Datei
input string       InpFileName="Close.bin"; // der Dateiname
input string       InpDirectoryName="Data"; // der Verzeichnisname
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    datetime date_finish=TimeCurrent();
    double   close_buff[];
    datetime time_buff[];
    int      size;
//--- stürzen Sie den Fehlerwert
    ResetLastError();
//--- Kopieren Sie der Schlußpreis für jede Bar
    if(CopyClose(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,close_buff)==-1)
    {
        PrintFormat("Fehler beim Kopieren von Werte des Schlußpreises. Fehlercode = %d",
            GetLastError());
        return;
    }
//--- Kopieren Sie die Zeit für jede Bar
    if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
    {
        PrintFormat("Fehler beim Kopieren von Zeitwerte. Fehlercode = %d",GetLastError());
        return;
    }
//--- erhalten Sie die Größe des Buffers
    size=ArraySize(close_buff);
//--- öffnen Sie die Datei zum Schreiben die Werte (wenn nicht, dann sie automatisch erstellen)
    ResetLastError();
    int file_handle=FileOpen(InpDirectoryName+"\\"+InpFileName,FILE_READ|FILE_WRITE|FILE_CREATE);
    if(file_handle!=INVALID_HANDLE)
    {
        PrintFormat("%s Datei ist zum Lesen geöffnet",InpFileName);
        PrintFormat("Pfad zur Datei: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
        //--- zeichnen Sie die Zeit und die Werte der Schlußpreis in die Datei auf
        for(int i=0;i<size;i++)
        {
            FileWriteDouble(file_handle,(double)time_buff[i]);
            FileWriteFloat(file_handle,(float)close_buff[i]);
        }
        //--- schließen Sie die Datei
        FileClose(file_handle);
        PrintFormat("Die Daten sind aufgezeichnet, die Datei %s geschlossen",InpFileName);
    }
    else
        PrintFormat("Fehler beim Öffnen der Datei %s, Fehlercode = %d",InpFileName,GetLastError());
}

```

```
}
```

Sehen Sie auch

[Realtypen \(double, float\)](#), [FileWriteDouble](#)

## FileWriteInteger

Schreibt in eine bin-Datei den Wert des Parameters des Typs int von der angegebenen Position des Dateianzeigers.

```
uint FileWriteInteger(  
    int file_handle,      // Datei-Handle  
    int value,            // geschriebener Wert  
    int size=INT_VALUE    // Größe in Bytes  
);
```

### Parameter

*file\_handle*

[in] Dateiattribut, das durch die Funktion [FileOpen\(\)](#) zurückgegeben wird.

*value*

[in] Ganzzahliger Wert.

*size=INT\_VALUE*

[in] Anzahl der Bytes (bis 4 einschliesslich), die geschrieben werden müssen. Entsprechende Konstanten werden vorausgesehen: CHAR\_VALUE=1, SHORT\_VALUE=2 und INT\_VALUE=4, so kann die Funktion den ganzzahligen Wert des Typs char, uchar, short, ushort, int oder uint schreiben.

### Rückgabewert

Im Erfolgsfall gibt die Funktion die Anzahl der geschriebenen Bytes zurück. Dateianzeiger wird um dieselbe Anzahl der Bytes verschoben.

### Beispiel:

```
//+-----+
//|                                     Demo_FileWriteInteger.mq5 |
//|                                     Copyright 2013, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- Beim Starten des Scripts zeigen Sie den Fenster mit den Eingabeparametern an
#property script_show_inputs
//--- die Parameter für das Erhalten der Daten aus dem Terminal
input string      InpSymbolName="EURUSD";           // Währungspaar
input ENUM_TIMEFRAMES InpSymbolPeriod=PERIOD_H1;    // Zeitspanne
input datetime     InpDateStart=D'2013.01.01 00:00'; // das Anfangsdatum zum Kop
//--- die Parameter für die Aufzeichnung der Daten in die Datei
input string       InpFileName="Trend.bin"; // der Dateiname
input string       InpDirectoryName="Data"; // der Verzeichnisname
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    datetime date_finish=TimeCurrent();
    double   close_buff[];
    datetime time_buff[];
    int      size;
//--- stürzen Sie den Fehlerwert
    ResetLastError();
//--- Kopieren Sie der Schlußpreis für jede Bar
    if(CopyClose(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,close_buff)==-1)
    {
        PrintFormat("Fehler beim Kopieren von Werte des Schlußpreises. Fehlercode = %d",
            GetLastError());
        return;
    }
//--- Kopieren Sie die Zeit für jede Bar
    if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
    {
        PrintFormat("Fehler beim Kopieren von Zeitwerte. Fehlercode = %d",GetLastError());
        return;
    }
//--- erhalten Sie die Größe des Buffers
    size=ArraySize(close_buff);
//--- öffnen Sie die Datei zum Schreiben die Werte (wenn nicht, dann sie automatisch e
    ResetLastError();
    int file_handle=FileOpen(InpDirectoryName+"\\"+InpFileName,FILE_READ|FILE_WRITE|FI
    if(file_handle!=INVALID_HANDLE)
    {
        PrintFormat("%s Datei ist zum Lesen geöffnet",InpFileName);
        PrintFormat("Pfad zur Datei: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH)
        //---
        int up_down=0; // Die Fahne der Tendenz
        int arr_size; // die Größe des Arrays arr
        uchar arr[]; // das Array vom Typ uchar
        //--- schreiben Sie die Zeitwerte in die Datei
        for(int i=0;i<size-1;i++)
        {
            //--- Vergleichen Sie die Schlußpreis an den aktuellen und folgenden Bars
            if(close_buff[i]<=close_buff[i+1])
            {
                if(up_down!=1)
                {

```

```

        //--- schreiben Sie den Datumswert in einer Datei mit FileWriteInteger
        StringToCharArray(TimeToString(time_buff[i]),arr);
        arr_size=ArraySize(arr);
        //--- schreiben Sie zuerst die Anzahl der Symbolen im Array
        FileWriteInteger(file_handle,arr_size,INT_VALUE);
        //--- schreiben Sie die Symbole selbst
        for(int j=0;j<arr_size;j++)
            FileWriteInteger(file_handle,arr[j],CHAR_VALUE);
        //--- ändern Sie die Fahne der Tendenz
        up_down=1;
    }
}
else
{
    if(up_down!=-1)
    {
        //--- schreiben Sie den Datumswert in einer Datei mit FileWriteInteger
        StringToCharArray(TimeToString(time_buff[i]),arr);
        arr_size=ArraySize(arr);
        //--- schreiben Sie zuerst die Anzahl der Symbolen im Array
        FileWriteInteger(file_handle,arr_size,INT_VALUE);
        //--- schreiben Sie die Symbole selbst
        for(int j=0;j<arr_size;j++)
            FileWriteInteger(file_handle,arr[j],CHAR_VALUE);
        //--- ändern Sie die Fahne der Tendenz
        up_down=-1;
    }
}
//--- schließen Sie die Datei
FileClose(file_handle);
PrintFormat("Die Daten sind aufgezeichnet, die Datei %s geschlossen",InpFileName);
}
else
    PrintFormat("Fehler beim Öffnen der Datei %s, Fehlercode = %d",InpFileName,GetLastError());
}

```

Sehen Sie auch

[IntegerToString](#), [StringToInteger](#), [Ganzzahlige Typen](#)



## FileWriteLong

Schreibt den Parameterwert des Typs long in eine bin-Datei von der laufenden Position des Dateianzeigers.

```
uint FileWriteLong(  
    int    file_handle,    // Datei-Handle  
    long   value           // der zu geschriebene Wert  
);
```

### Parameter

*file\_handle*

[in] Dateiattribut, das durch die Funktion [FileOpen\(\)](#) zurückgegeben wird.

*value*

[in] Wert des Typs long.

### Rückgabewert

Im Erfolgsfall gibt die Funktion Anzahl der geschriebenen Bytes zurück (in diesem Fall [sizeof\(long\)](#) =8). Dateianzeiger wird um dieselbe Anzahl der Bytes verschoben.

### Beispiel:

```
//+-----+
//|                                     Demo_FileWriteLong.mq5 |
//|                                     Copyright 2013, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- Beim Starten des Scripts zeigen Sie den Fenster mit den Eingabeparametern an
#property script_show_inputs
//--- die Parameter für das Erhalten der Daten aus dem Terminal
input string      InpSymbolName="EURUSD";           // Währungspaar
input ENUM_TIMEFRAMES InpSymbolPeriod=PERIOD_H1;    // Zeitspanne
input datetime     InpDateStart=D'2013.01.01 00:00'; // das Anfangsdatum zum Kopieren
//--- die Parameter für die Aufzeichnung der Daten in die Datei
input string       InpFileName="Volume.bin";        // der Dateiname
input string       InpDirectoryName="Data";        // der Verzeichnisname
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    datetime date_finish=TimeCurrent();
    long      volume_buff[];
    datetime  time_buff[];
    int       size;
//--- stürzen Sie den Fehlerwert
    ResetLastError();
//--- kopieren Sie die Tick Volumen für jede Bar
    if(CopyTickVolume(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,volume_buff))
    {
        PrintFormat("Fehler beim Kopieren die Werte von Tick-Volumen. Fehlercode = %d",GetLastError());
        return;
    }
//--- Kopieren Sie die Zeit für jede Bar
    if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
    {
        PrintFormat("Fehler beim Kopieren von Zeitwerte. Fehlercode = %d",GetLastError());
        return;
    }
//--- erhalten Sie die Größe des Puffers
    size=ArraySize(volume_buff);
//--- öffnen Sie die Datei zum Schreiben die Indikatorwerte (wenn nicht, dann sie automatisch erstellen)
    ResetLastError();
    int file_handle=FileOpen(InpDirectoryName+"\\"+InpFileName,FILE_READ|FILE_WRITE|FILE_CREATE);
    if(file_handle!=INVALID_HANDLE)
    {
        PrintFormat("%s Datei ist zum Lesen geöffnet",InpFileName);
        PrintFormat("Pfad zur Datei: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
        //--- schreiben Sie zuerst die Größe der Beispieldaten
        FileWriteLong(file_handle,(long)size);
        //--- schreiben Sie die Zeit und die Volumenwerte in die Datei
        for(int i=0;i<size;i++)
        {
            FileWriteLong(file_handle,(long)time_buff[i]);
            FileWriteLong(file_handle,volume_buff[i]);
        }
        //--- schließen Sie die Datei
        FileClose(file_handle);
        PrintFormat("Die Daten sind aufgezeichnet, die Datei %s geschlossen",InpFileName);
    }
}
```

```
else  
    PrintFormat("Fehler beim Öffnen der Datei %s, Fehlercode = %d",InpFileName,GetLe  
}
```

Sehen Sie auch

[Ganzzahlige Typen](#), [FileWriteInteger](#)

## FileWriteString

Zeichnet in die Datei des Typs BIN, CSV oder TXT, die Wert des Parameters des Typs string von der laufenden Position des Dateianzeigers auf. Bei der Aufzeichnung in die Datei des Typs CSV oder TXT, wenn in der Zeile das Symbol '\n' (LF) ohne vorangehendes Symbol '\r' (CR) anwesend ist, so wird vor dem Symbol '\n' das fehlende Symbol '\r' fertiggeschrieben.

```
uint FileWriteString(  
    int          file_handle,    // Datei-Handle  
    const string text_string,    // die geschriebene Zeile  
    int          length=-1      // Anzahl der Symbole  
);
```

### Parameter

*file\_handle*

[in] Dateiattribut, das durch die Funktion [FileOpen\(\)](#) zurückgegeben wird.

*text\_string*

[in] Zeile.

*length=-1*

[in] Anzahl der Symbole, die geschrieben werden müssen. Parameter ist für Schreiben der Zeile in die Datei des Typs BIN notwendig. Wenn die Größe nicht angegeben wird, wird die ganze Zeile ohne terminale 0 geschrieben. Wenn die angegebene Größe weniger als Zeilenlänge ist, wird ein Teil der Zeile ohne terminale 0 geschrieben. Wenn die angegebene Größe mehr als die Zeilenlänge ist, wird die Zeile durch entsprechende Anzahl der Nullen ergänzt. Für die Dateien des Typs CSV oder TXT wird dieser Parameter auch die Zeile ignoriert schreibt sich vollständig ein.

### Rückgabewert

Im Erfolgsfall gibt die Funktion die Anzahl der geschriebenen Bytes zurück. Dateianzeiger wird um dieselbe Anzahl der Bytes verschoben.

### Bemerkung

Es ist bemerkenswert, dass beim Schreiben in die Datei, die durch die Flagge [Flagge](#) FILE\_UNICODE (oder ohne Flagge FILE\_ANSI) geöffnet wurde, wird die Anzahl der geschriebenen Bytes um das Doppelte mehr als die Anzahl der geschriebenen Zeilensymbole. Beim Schreiben in die Datei, die durch die Flagge FILE\_ANSI eröffnet wurde, wird die Anzahl der geschriebenen Bytes mit der Anzahl der geschriebenen Zeilensymbole zusammenfallen.

### Beispiel:

```

//+-----+
//|                                     Demo_FileWriteString.mq5 |
//|                                     Copyright 2013, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- Beim Starten des Scripts zeigen Sie den Fenster mit den Eingabeparametern an
#property script_show_inputs
//--- die Parameter für das Erhalten der Daten aus dem Terminal
input string      InpSymbolName="EURUSD";           // Währungspaar
input ENUM_TIMEFRAMES InpSymbolPeriod=PERIOD_H1;    // Zeitspanne
input int          InpMAPeriod=14;                  // die durchschnittliche Periode
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE; // der Typ des Preises
input datetime     InpDateStart=D'2013.01.01 00:00'; // das Anfangsdatum zum Kopieren
//--- die Parameter für die Aufzeichnung der Daten in die Datei
input string       InpFileName="RSI.csv";           // der Dateiname
input string       InpDirectoryName="Data";         // der Verzeichnisname
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    datetime date_finish; // das Enddatum zum Kopieren von Daten
    double    rsi_buff[]; // ein Array von Werten des Indikators
    datetime  date_buff[]; // ein Array von Daten des Indikators
    int       rsi_size=0; // die Größe der Array von Indikator
//--- die Zeit des Abschlusses - laufend
    date_finish=TimeCurrent();
//--- bekommen Sie die Handle des Indikators RSI
    ResetLastError();
    int rsi_handle=iRSI(InpSymbolName,InpSymbolPeriod,InpMAPeriod,InpAppliedPrice);
    if(rsi_handle==INVALID_HANDLE)
    {
        //--- es misslang, die Handle des Indikators zu bekommen
        PrintFormat("Fehler beim Erhalten der Handle des Indikators. Fehlercode = %d",GetLastError());
        return;
    }
//--- befinden Sie sich im Zyklus, bis der Indikator alle Werten rechnen wird
    while(BarsCalculated(rsi_handle)==-1)
        Sleep(10); // die Verzögerungszeit, damit der Indikator dazugekommen ist die Werte
//--- kopieren Sie die Werte eines Indikators für eine bestimmte Periode
    ResetLastError();
    if(CopyBuffer(rsi_handle,0,InpDateStart,date_finish,rsi_buff)==-1)
    {
        PrintFormat("Fehler beim Kopieren der Werte des Indikators. Fehlercode = %d",GetLastError());
        return;
    }
//--- kopieren Sie die entsprechende Zeit für die Werte des Indikators
    ResetLastError();
    if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,date_buff)==-1)
    {
        PrintFormat("Fehler beim Kopieren von Zeitwerten. Fehlercode = %d",GetLastError());
        return;
    }
//--- befreien Sie das Gedächtnis, das mit dem Indikator eingenommen wird
    IndicatorRelease(rsi_handle);
//--- erhalten Sie die Größe des Puffers
    rsi_size=ArraySize(rsi_buff);
//--- öffnen Sie die Datei zum Schreiben die Indikatorwerte (wenn nicht, dann sie auto

```

```

ResetLastError();
int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FILE_APPEND);
if(file_handle!=INVALID_HANDLE)
{
    PrintFormat("%s Datei ist zum Lesen geöffnet",InpFileName);
    PrintFormat("Pfad zur Datei: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
    //--- bereiten Sie die Hilfsvariablen vor
    string str="";
    bool is_formed=false;
    //--- schreiben Sie das Datum auf die überkauft und überverkauft Zonen
    for(int i=0;i<rsi_size;i++)
    {
        //--- die Prüfung der Werten des Indikators
        if(rsi_buff[i]>=70 || rsi_buff[i]<=30)
        {
            //--- der erste Wert in dieser Zone
            if(!is_formed)
            {
                //--- fügen Sie den Wert und das Datum ein
                str=(string)rsi_buff[i]+"\\t"+(string)date_buff[i];
                is_formed=true;
            }
            else
                str+="\\t"+(string)rsi_buff[i]+"\\t"+(string)date_buff[i];
            //--- der Übergang auf die nächste Iteration des Zyklus
            continue;
        }
        //--- die Prüfung der Fahne
        if(is_formed)
        {
            //---die Zeile ist gebildet, wir werden sie in die Datei aufzeichnen
            FileWriteString(file_handle,str+"\\r\\n");
            is_formed=false;
        }
    }
    //--- schließen Sie die Datei
    FileClose(file_handle);
    PrintFormat("Die Daten sind aufgezeichnet, die Datei %s geschlossen",InpFileName);
}
else
    PrintFormat("Fehler beim Öffnen der Datei %s, Fehlercode = %d",InpFileName,GetLastError());
}

```

Sehen Sie auch

[Typ string](#), [StringFormat](#)

## FileWriteStruct

Schreibt in die bin-Datei Inhalt der Struktur, die als Parameter von der laufenden Position des Dateianzeigers übertragen wurde.

```
uint FileWriteStruct(  
    int          file_handle,      // Datei-Handle  
    const void&  struct_object,   // Referenz auf das Objekt  
    int          size=-1          // die aufzugezeichnete Größe in Bytes  
);
```

### Parameter

*file\_handle*

[in] Dateidescriptor, der durch die Funktion [FileOpen\(\)](#) zurückgegeben wird.

*struct\_object*

[in] Referenz auf das Objekt der angegebenen Struktur. Struktur muss keine Zeilen, [dynamische Felder](#) und [virtuelle Funktionen](#) enthalten.

*size=-1*

[in] Anzahl der Bytes, die geschrieben werden müssen. Wenn die Größe nicht angegeben wird oder die angegebene Anzahl der Bytes mehr als die Strukturgröße ist, wird die ganze Struktur vollstaendig geschrieben.

### Rückgabewert

Im Erfolgsfall gibt die Funktion die Anzahl der geschriebenen Bytes zurück. Dateianzeiger wird um dieselbe Anzahl der Bytes verschoben.

### Beispiel:

```
//+-----+
//|                                     Demo_FileWriteStruct.mq5 |
//|                                     Copyright 2013, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2013, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- Beim Starten des Scripts zeigen Sie den Fenster mit den Eingabeparametern an
#property script_show_inputs
//--- die Parameter für das Erhalten der Daten aus dem Terminal
input string      InpSymbolName="EURUSD";           // Währungspaar
input ENUM_TIMEFRAMES InpSymbolPeriod=PERIOD_H1;    // Zeitspanne
input datetime     InpDateStart=D'2013.01.01 00:00'; // das Anfangsdatum zum Kopieren
//--- die Parameter für die Aufzeichnung der Daten in die Datei
input string       InpFileName="EURUSD.txt";        // der Dateiname
input string       InpDirectoryName="Data";         // der Verzeichnisname
//+-----+
//| Struktur für die Speicherung von Daten-Kerzen |
//+-----+
struct candlesticks
{
    double      open; // Eröffnungspreis
    double      close; // Schlußpreis
    double      high; // der maximale Preis
    double      low; // der minimale Preis
    datetime     date; // Datum
};
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    datetime     date_finish=TimeCurrent();
    int          size;
    datetime     time_buff[];
    double       open_buff[];
    double       close_buff[];
    double       high_buff[];
    double       low_buff[];
    candlesticks cand_buff[];
//--- stürzen Sie den Fehlerwert
    ResetLastError();
//---erhalten Sie die Zeit des Erscheinens der Bars aus dem Umfang
    if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
    {
        PrintFormat("Fehler beim Kopieren der Zeitwerte. Fehlercode = %d",GetLastError());
        return;
    }
//--- erhalten Sie die maximalen Preise der Bars aus dem Umfang
    if(CopyHigh(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,high_buff)==-1)
    {
        PrintFormat("Fehler beim Kopieren der maximalen Preise. Fehlercode = %d",GetLastError());
        return;
    }
//--- erhalten Sie die minimalen Preise der Bars aus dem Umfang
    if(CopyLow(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,low_buff)==-1)
    {
        PrintFormat("Fehler beim Kopieren der minimalen Preise. Fehlercode = %d",GetLastError());
        return;
    }
}
```



```

//--- erhalten Sie die Eröffnungspreise der Bars aus dem Umfang
if(CopyOpen(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,open_buff)==-1)
{
    PrintFormat("Fehler beim Kopieren von Werte des Eröffnungspreises. Fehlercode = %d", GetLastError());
    return;
}
//--- erhalten Sie die Schlußpreise der Bars aus dem Umfang
if(CopyClose(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,close_buff)==-1)
{
    PrintFormat("Fehler beim Kopieren von Werte des Schlußpreises. Fehlercode = %d", GetLastError());
    return;
}
//--- bestimmen Sie die Dimension des Arrays
size=ArraySize(time_buff);
//--- Speichern Sie alle Daten in der Arraystruktur
ArrayResize(cand_buff,size);
for(int i=0;i<size;i++)
{
    cand_buff[i].open=open_buff[i];
    cand_buff[i].close=close_buff[i];
    cand_buff[i].high=high_buff[i];
    cand_buff[i].low=low_buff[i];
    cand_buff[i].date=time_buff[i];
}

//--- öffnen Sie die Datei zum Schreiben des Arrays der Struktur in die Datei (wenn nicht existiert)
ResetLastError();
int file_handle=FileOpen(InpDirectoryName+"\\\\"+InpFileName,FILE_READ|FILE_WRITE|FILE_APPEND);
if(file_handle!=INVALID_HANDLE)
{
    PrintFormat("%s Datei ist zum Lesen geöffnet",InpFileName);
    PrintFormat("Pfad zur Datei: %s\\Files\\",TerminalInfoString(TERMINAL_COMMONDATA_PATH));
    //--- bereiten Sie den Zähler der Zahl der Bytes vor
    uint counter=0;
    //--- zeichnen Sie die Werte des Arrays im Zyklus
    for(int i=0;i<size;i++)
        counter+=FileWriteStruct(file_handle,cand_buff[i]);
    PrintFormat("In die Datei %s ist es %d das Byte der Informationen aufgezeichnet",InpFileName,counter);
    PrintFormat("Die Gesamtanzahl der Bytes: %d * %d * %d = %d, %s",size,5,8,size*5*8);
    //--- schließen Sie die Datei
    FileClose(file_handle);
    PrintFormat("Die Daten sind aufgezeichnet, die Datei %s geschlossen",InpFileName);
}
else
    PrintFormat("Fehler beim Öffnen der Datei %s, Fehlercode = %d",InpFileName,GetLastError());
}

```

Sehen Sie auch

[Strukturen und Klassen](#)

## FolderCreate

Erzeugt ein Verzeichnis im Verzeichnis Files (abhängig vom Wert common\_flag)

```
bool FolderCreate(  
    string  folder_name,      // Zeile mit dem Namen des neuen Ordners  
    int     common_flag=0    // Geltungsbereich  
);
```

### Parameter

*folder\_name*

[in] der Name des Verzeichnisses, das erzeugt werden muss. Enthält den ganzen Pfad zum Ordner.

*common\_flag=0*

[in] [Flagge](#), die die Lage des Verzeichnisses bestimmt. Wenn common\_flag=FILE\_COMMON ist, befindet sich das Verzeichnis im Gesamtordner aller Kundenterminals \Terminal\Common\Files. Anderenfalls befindet sich das Verzeichnis im lokalen Ordner (MQL5\files oder MQL5\tester\files beim Testen).

### Rückgabewert

Gibt true im Erfolgsfall zurück, anderenfalls false.

### Bemerkung

Aus Sicherheitsgründen ist die Arbeit mit Dateien in MQL5 streng gesteuert. Dateien, mit denen Datei-Operationen mittels MQL5 durchgeführt werden, können nicht außerhalb der Datei-Sandbox sein.

### Beispiel:

```

//+-----+
//|                                     Demo_FolderCreate.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://team.metaquotes.ru/email/view/599588 |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://team.metaquotes.ru/email/view/599588"
#property version   "1.00"
//--- Beschreibung
#property description "Das Skript zeigt ein Beispiel der Funktion FolderCreate()."
#property description "Ein Externer Parameter definiert das Verzeichnis zum Erstellen"
#property description "Nach dem Ausführen des Skripts wird ein Ordner-Struktur erstellt"

//--- Beim Starten des Scripts zeigen Sie den Fenster mit den Eingabeparametern an
#property script_show_inputs
//--- Der Eingabeparameter definiert den Ordner, in dem das Script läuft
input bool    common_folder=false; // Gemeinsamer Ordner von allen Terminals
int        flag=0;                // Flag-Wert bestimmt den Speicherort der Datei-Op

//+-----+
//| Script program start function                                     |
//+-----+
void OnStart()
{
    string working_folder;
    //--- Setzen Sie den Flag-Wert, wenn der externe Parameter common_folder==true
    if(common_folder)
    {
        flag=FILE_COMMON;
        //--- Finden Sie in welchem Ordner wir arbeiten
        working_folder=TerminalInfoString(TERMINAL_COMMONDATA_PATH)+"\\MQL5\\Files";
    }
    else working_folder=TerminalInfoString(TERMINAL_DATA_PATH)+"\\MQL5\\Files";
    //--- Der Ordner, der in Verzeichnis MQL5\\Files erstellt wird
    string root="Folder_A";
    if(CreateFolder(working_folder,root,flag))
    {
        //--- Erstellen Sie einen Unterordner in ihm Child_Folder_B1
        string folder_B1="Child_Folder_B1";
        string path=root+"\\ "+folder_B1; // Erstellen Sie einen Ordner-Namen be
        if(CreateFolder(working_folder,path,flag))
        {
            //--- In diesem Ordner erstellen Sie 3 anderen Unterordner
            string folder_C11="Child_Folder_C11";
            string child_path=path+"\\ "+folder_C11; // Erstellen Sie einen Ordner-Namen be
            CreateFolder(working_folder,child_path,flag);
            //--- Der zweite Unterordner
            string folder_C12="Child_Folder_C12";
            child_path=path+"\\ "+folder_C12;
            CreateFolder(working_folder,child_path,flag);

            //--- Der dritte Unterordner
            string folder_C13="Child_Folder_C13";
            child_path=path+"\\ "+folder_C13;
            CreateFolder(working_folder,child_path,flag);
        }
    }
}
//---
}
//+-----+
//| Versuchen Sie einen Ordner zu erstellen und zeigen die Meldungen |
//+-----+

```

```
bool CreateFolder(string working_folder,string folder_path,int file_flag)
{
    //--- Debug-Meldung
    PrintFormat("folder_path=%s",folder_path);
    //--- Versuch, einen Ordner auf Pfad MQL5\Files zu erstellen
    if(FolderCreate(folder_path,file_flag))
    {
        //--- Zeigen Sie den vollen Pfad zum erstellten Ordner
        PrintFormat("Ordner %s wurde erstellt",working_folder+"\\ "+folder_path);
        //--- Fehlercode rücksetzen
        ResetLastError();
        //--- Erfolg zurückgeben
        return true;
    }
    else
        PrintFormat("Ordner %s konnte nicht erstellt werden. Fehlercode %d",working_folder,file_flag);
    //--- Versagen
    return false;
}
```

Sehen Sie auch

[FileOpen\(\)](#), [FolderClean\(\)](#), [FileCopy\(\)](#)

## FolderDelete

Entfernt das angegeben Verzeichnis. Wenn der Ordner leer ist, kann er nicht entfernt werden.

```
bool FolderDelete(  
    string  folder_name,      // Zeile mit dem Namen des entfernten Ordners  
    int     common_flag=0    // Geltungsbereich  
);
```

### Parameter

*folder\_name*

[in] Name des Verzeichnisses, das entfernt werden muss. Hat den ganzen Pfad zum Ordner.

*common\_flag=0*

[in] [Flagge](#), die die Lage des Verzeichnisses bestimmt. Wenn common\_flag=FILE\_COMMON, befindet sich das Verzeichnis im Gesamtordner aller Kundenterminals \Terminal\Common\Files. Anderenfalls befindet sich das Verzeichnis im lokalen Ordner (MQL5\files oder MQL5\tester\files beim Testen).

### Rückgabewert

Gibt true im Erfolgsfall zurück, anderenfalls false.

### Bemerkung

Aus Sicherheitsgründen ist die Arbeit mit Dateien in MQL5 streng gesteuert. Dateien, mit denen Datei-Operationen mittels MQL5 durchgeführt werden, können nicht außerhalb der Datei-Sandbox sein.

Wenn es im Verzeichnis wenigstens eine Datei und / oder ein Unterverzeichnis gibt, ist die Entfernung solches Verzeichnisses unmöglich, es muss vorerst abgeklärt werden. durch die Funktion [FolderClean\(\)](#) kann der Ordner von allen Dateien und verschachtelten Unterordnern abgeklärt werden.

### Beispiel:

```
//+-----+
//|                                     Demo_FolderDelete.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://team.metaquotes.ru/email/view/599588 |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://team.metaquotes.ru/email/view/599588"
#property version   "1.00"
//--- Beschreibung
#property description "Das Skript zeigt ein Beispiel der Funktion FolderDelete()."
#property description "Erstellen Sie zunächst zwei Ordner, einer ist leer und der andere enthält eine Datei."
#property description "Beim Versuch einen nicht-leeren Ordner zu löschen, erhalten wir eine Fehlermeldung."

//--- Beim Starten des Scripts zeigen Sie den Fenster mit den Eingabeparametern an
#property script_show_inputs
//--- Eingabeparameter
input string  firstFolder="empty";    // Leerer Ordner
input string  secondFolder="nonempty"; // Ein Ordner mit einer Datei
string filename="delete_me.txt";      // der Dateiname, die Sie im Ordner secondFolder erstellen

//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
//--- Schreiben Sie die Datei-Handle hier
    int handle;
//--- Finden Sie in welchem Ordner Sie arbeiten
    string working_folder=TerminalInfoString(TERMINAL_DATA_PATH)+"\\MQL5\\Files";
//--- Debug-Meldung
    PrintFormat("working_folder=%s",working_folder);
//--- Versuchen Sie einen leeren Ordner auf Pfad MQL5\\Files zu erstellen
    if(FolderCreate(firstFolder,0)) // 0 bedeutet, dass Sie in lokalem Ordner des Terminals arbeiten
    {
        //--- Zeigen Sie den vollen Pfad zum erstellten Ordner
        PrintFormat("Ordner %s wurde erstellt",working_folder+"\\ "+firstFolder);
        //--- Fehlercode rücksetzen
        ResetLastError();
    }
    else
        PrintFormat("Ordner %s konnte nicht erstellt werden. Fehlercode %d",working_folder,GetLastError());

//--- Erstellen Sie nun einen nicht-leeren mit der FileOpen()-Funktion
    string filepath=secondFolder+"\\ "+filename; // Formen Sie den Pfad zur Datei, die Sie erstellen
    handle=FileOpen(filepath,FILE_WRITE|FILE_TXT); // Flag FILE_WRITE ist in diesem Fall nicht erforderlich
    if(handle!=INVALID_HANDLE)
        PrintFormat("Datei %s wurde für lesen eröffnet",working_folder+"\\ "+filepath);
    else
        PrintFormat("Datei %s konnte nicht im Ordner %s erstellt werden. Fehlercode=%d",filename,working_folder,GetLastError());

    Comment(StringFormat("Vorbereiten auf Löschung der Dateien %s und %s", firstFolder, secondFolder));
//--- Eine kurze Pause von 5 Sekunden, so dass Sie die Meldung auf Chart lesen können
    Sleep(5000); // Sleep() kann nicht in den Indikatoren verwendet werden!

//--- Im Dialogfenster fragen Sie den Benutzer
    int choice=MessageBox(StringFormat("Möchten Sie Ordner %s und %s löschen?", firstFolder, secondFolder),
                          "Löschung der Ordner",
                          MB_YESNO|MB_ICONQUESTION); // Zwei Schaltflächen - "Yes" und "No"

//--- Aktionen je nach der gewählten Option
    if(choice==IDYES)
    {

```

```
//--- Den Kommentar aus dem Chart löschen
Comment("");
//--- Eine Meldung in "Experts" Journal hinzufügen
PrintFormat("Versuch die Ordner %s und %s zu löschen",firstFolder, secondFolder);
ResetLastError();
//--- Löschen den leeren Ordner
if(FolderDelete(firstFolder))
    //--- Diese Meldung soll erscheinen, weil der Ordner leer ist
    PrintFormat("Ordner %s wurde erfolgreich gelöscht",firstFolder);
else
    PrintFormat("Ordner %s konnte nicht gelöscht werden. Fehlercode=%d", firstFolder, GetLastError());

ResetLastError();
//--- Löschen Sie den Ordner mit einer Datei
if(FolderDelete(secondFolder))
    PrintFormat("Ordner %s wurde erfolgreich gelöscht", secondFolder);
else
    //--- Diese Meldung soll erscheinen, weil der Ordner hat eine Datei
    PrintFormat("Ordner %s konnte nicht gelöscht werden. Fehlercode=%d", secondFolder, GetLastError());
}
else
    Print("Löschung wurde abgebrochen");
//---
}
```

Sehen Sie auch

[FileOpen\(\)](#), [FolderClean\(\)](#), [FileMove\(\)](#)

## FolderClean

Entfernt alle Dateien im angegebenen Ordner.

```
bool FolderClean(  
    string folder_name,      // Zeile mit dem Ordernamen  
    int common_flag=0        // Geltungsbereich  
);
```

### Parameter

*folder\_name*

[in] Name des Verzeichnisses, in dem alle Dateien entfernt werden müssen. Enthält den gesamten Pfad zum Ordner.

*common\_flag=0*

[in] [Flagge](#), die die Lage des Verzeichnisses bestimmt. Wenn common\_flag=FILE\_COMMON ist, befindet sich das Verzeichnis im Gesamtfolder aller Kundenterminals \Terminal\Common\Files. Anderenfalls befindet sich das Verzeichnis im lokalen Ordner (MQL5\files oder MQL5\tester\files beim Testen).

### Rückgabewert

Gibt true im Erfolgsfall zurück, anderenfalls false.

### Bemerkung

Aus Sicherheitsgründen ist die Arbeit mit Dateien in MQL5 streng gesteuert. Dateien, mit denen Datei-Operationen mittels MQL5 durchgeführt werden, können nicht außerhalb der Datei-Sandbox sein.

Verwenden Sie diese Funktion vorsichtig, denn alle Dateien und verschachtelte Unterverzeichnisse werden unwiederbringlich entfernt.

### Beispiel:



```

//+-----+
//|                                     Demo_FolderClean.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://team.metaquotes.ru/email/view/599588 |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://team.metaquotes.ru/email/view/599588"
#property version   "1.00"
//--- Beschreibung
#property description "Das Skript zeigt ein Beispiel der Funktion FolderClean()."
#property description "Zuerst erstellen Sie Dateien in einem bestimmten Ordner mit Fil
#property description "Dann, bevor Löschung der Dateien, erscheint die Warnung Message

//--- Beim Starten des Scripts zeigen Sie den Fenster mit den Eingabeparametern an
#property script_show_inputs
//--- Eingabeparameter
input string  foldername="demo_folder"; // Erstellen wir einen Ordner in MQL5/Files,
input int     files=5;                  // Wie viele Dateien erstellen und löschen s
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    string name="testfile";
//--- Zuerst erstellen oder öffnen Sie Dateien im Datenordner des Terminals
    for(int N=0;N<files;N++)
    {
        //--- Dateiname in der Form von 'demo_folder\testfileN.txt'
        string filemane=StringFormat("%s\\%s%d.txt",foldername,name,N);
        //--- Öffnen Sie eine Datei mit dem Flag zu schreiben, in diesem Fall wird Ordner
        int handle=FileOpen(filemane,FILE_WRITE);
        //--- Finden Sie, ob FileOpen() erfolgreich war
        if(handle==INVALID_HANDLE)
        {
            PrintFormat("Datei %s konnte nicht erstellt werden. Fehlercode ",filemane,Get
            ResetLastError();
        }
        else
        {
            PrintFormat("Datei %s wurde erfolgreich geöffnet",filemane);
            //--- Die offene Datei brauchen Sie nicht mehr, so schließen Sie sie
            FileClose(handle);
        }
    }

//--- Überprüfen Sie, wie viele Dateien der Ordner hat
    int k=FilesInFolder(foldername+"\\*.txt",0);
    PrintFormat("%d Dateien im Ordner %s gefunden",foldername,k);

//--- Im Dialogfenster fragen Sie den Benutzer
    int choice=MessageBox(StringFormat("Sie sind dabei, %d Dateien aus dem Ordner %s zu
        "Löschung der Dateien aus dem Ordner",
        MB_YESNO|MB_ICONQUESTION); // Zwei Schaltflächen - "Yes" und

    ResetLastError();

//--- Aktionen je nach der gewählten Option
    if(choice==IDYES)
    {
        //--- Sie beginnen zu löschen
        PrintFormat("Der Versuch, alle Dateien aus dem Ordner %s zu löschen",foldername)
        if(FolderClean(foldername,0))
    }
}

```

```

        PrintFormat("Die Dateien wurden erfolgreich gelöscht, %d Dateien bleiben im Ordner %s",
                    foldername,
                    FilesInFolder(foldername+"\\*.*",0));
    else
        PrintFormat("Dateien konnten nicht aus dem Ordner %s gelöscht werden. Fehlercode: %d",
                    foldername, GetLastError());
    }
    else
        PrintFormat("Löschung wurde abgebrochen");
//---
}
//+-----+
//| Gibt die Anzahl der Dateien im Ordner zurück |
//+-----+
int FilesInFolder(string path,int flag)
{
    int count=0;
    long handle;
    string filename;
//---
    handle=FileFindFirst(path,filename,flag);
//--- Wenn eine Datei gefunden wird, dann suchen Sie nach anderen
    if(handle!=INVALID_HANDLE)
    {
        //--- Dateinamen zeigen
        PrintFormat("File %s wurde gefunden",filename);
        //--- Erhöhen Sie den Zähler der gefundenen Dateien/Ordner
        count++;
        //--- Beginnen Sie suchen nach allen Dateien/Ordnern
        while(FileFindNext(handle,filename))
        {
            PrintFormat("Datei %s wurde gefunden",filename);
            count++;
        }
        //--- Vergessen Sie nicht die Suche Handle zu schließen
        FileFindClose(handle);
    }
    else // Handle konnte nicht erhalten werden
    {
        PrintFormat("Suche nach Dateien im Ordner %s fehlgeschlagen",path);
    }
//--- Ergebnis zurückgeben
    return count;
}

```

Sehen Sie auch

[FileFindFirst](#), [FileFindNext](#), [FileFindClose](#)

## Benutzerindikatoren

Gruppe der Funktionen, die für die Erzeugung der Benutzerindikatoren verwendet werden. Diese Funktionen können nicht beim Schreiben von Scripts und Ratgebern verwendet werden.

| Funktion                            | Massnahme                                                                                                                    |
|-------------------------------------|------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">SetIndexBuffer</a>      | Verbindet den Indikatorpuffer mit dem eindimensionalen dynamischen Feld <a href="#">Feld</a> des Typs <a href="#">double</a> |
| <a href="#">IndicatorSetDouble</a>  | Gibt den Wert des Indikators vor, der den Typ <a href="#">double</a> hat                                                     |
| <a href="#">IndicatorSetInteger</a> | Gibt Eigenschaftswert des Indikators vor, der den Typ <a href="#">int</a> hat                                                |
| <a href="#">IndicatorSetString</a>  | Gibt Eigenschaftswert des Indikators vor, der den Typ <a href="#">string</a> hat                                             |
| <a href="#">PlotIndexSetDouble</a>  | Gibt Zeilenwert des Indikators vor, der den Typ <a href="#">double</a> hat                                                   |
| <a href="#">PlotIndexSetInteger</a> | Gibt Zeilenwert des Indikators vor, der den Typ <a href="#">int</a> hat                                                      |
| <a href="#">PlotIndexSetString</a>  | Gibt Zeilenwert des Indikators vor, der den Typ <a href="#">string</a> hat                                                   |
| <a href="#">PlotIndexGetInteger</a> | Gibt Zeilenwert des Indikators zurück, der den <a href="#">ganzzahligen</a> Typ hat                                          |

[Eigenschaften der Indikator](#) können mit Hilfe der Compiler-Direktiven oder mit Funktionen angegeben werden. Um dies besser zu verstehen, ist es empfehlenswert, dass Sie [Stile der Indikator in den Beispielen](#) studieren.

Alle notwendigen Berechnungen der Benutzerindikatoren müssen in der vorbestimmten Funktion [OnCalculate\(\)](#) untergeordnet werden. Wenn die kurze Aufrufform der Funktion [OnCalculate\(\)](#) der Art

```
int OnCalculate (const int rates_total, const int prev_calculated, const int begin, co
```

verwendet wird, enthält die Variable *rates\_total* den Wert der gesamten Anzahl der Elemente des Feldes *price[]*, das als Eingabe-Parameter für Berechnung der Anzeigerwerte übertragen wurde.

Parameter *prev\_calculated* - ist das Ergebnis der Durchführung der Funktion [OnCalculate\(\)](#) auf dem vorangehenden Level und ermöglicht sparsamen Algorithmus für Berechnung der Indikatorwerte zu gestalten. ZB wenn der laufende Wert *rates\_total*=1000, und *prev\_calculated*=999, dann ist es vielleicht genug, Berechnungen nur für einen Wert des Anzeigerpuffers zu machen.

Wenn die Information über die Größe des Eingabefeldes *price* unzugänglich wäre, würde es zur Notwendigkeit führen, Berechnungen für 1000 Werte jedes Indikatorpuffers zu machen. Beim ersten Aufruf der Funktion [OnCalculate\(\)](#) ist der Wert *prev\_calculated*=0. Wenn sich das Feld *price[]* irgendwie verändert hat, ist *prev\_calculated* in diesem Fall auch 0.

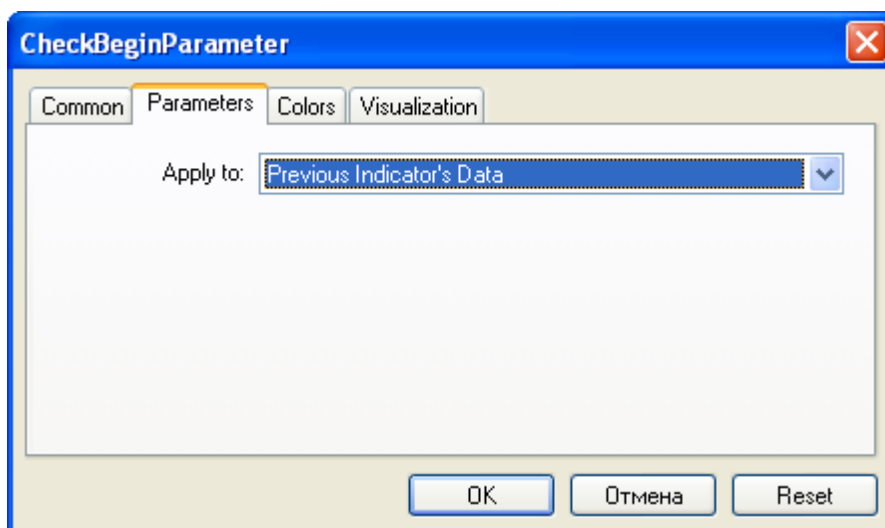
Parameter *begin* zeigt die Anzahl der Initialwerte des Feldes *price*, die keine Daten für die Berechnung

enthalten. ZB wenn als Eingabefeld die Werte des Indikators Accelerator Oscillator verwendet wurden (für den die ersten 37 Werte nicht berechnet werden), ist begin=37. Als Beispiel betrachten wir einen einfachen Anzeiger:

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots 1
//---- plot Label1
#property indicator_label1 "Label1"
#property indicator_type1 DRAW_LINE
#property indicator_color1 clrRed
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//--- indicator buffers
double Label1Buffer[];
//+-----+
//| Custom indicator initialization function |
//+-----+
void OnInit()
{
//--- indicator buffers mapping
SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
//---
; }
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const int begin,
               const double &price[])

{
//---
Print("begin = ",begin," prev_calculated = ",prev_calculated," rates_total = ",rates_total);
//--- return value of prev_calculated for next call
return(rates_total);
; }
```

Ziehen wir das aus dem Fenster "Navigator" ins Fenster des Indikators Accelerator Oscillator und geben wir an, dass Berechnungen auf der Grund der Werte des früheren Indikators durchgeführt werden:



Im Ergebnis wird der Wert `prev_calculated` beim ersten Aufruf der Funktion `OnCalculate()` 0 sein, bei weiteren Aufrufen wird der Wert dem Werten `rates_total` gleich sein (bis die Anzahl der Bars am Preischart größer wird).



Wert des Parameters `begin` wird der Anzahl der Initialbars gleich, für die Werte des Indikators Accelerator nicht berechnet werden auf Grund der Logik dieses Indikators. Wenn wir den Ausgangskode des Benutzerindikators `Accelerator.mq5` ansehen, sehen wir in der Funktion `OnInit()` die folgenden Zeilen:

```
//--- sets first bar from what index will be drawn
PlotIndexSetInteger(0, PLOT_DRAW_BEGIN, 37);
```

Gerade durch die Funktion `PlotIndexSetInteger(0, PLOT_DRAW_BEGIN, empty_first_values)` stellen wir die Anzahl der unbedeutenden ersten Werte im Nullindikatorfeld, die wir für die Berechnungen

(empty\_first\_values) nicht zu beachten brauchen. So haben wir Mechanismen um:

1. die Anzahl der Initialwerte des Indikators anzugeben, die für Berechnungen im anderen Benutzerindikator nicht zu verwendet sind;
2. die Information über die Anzahl der ersten Werte zu bekommen, die beim Aufruf eines anderen Benutzerindikatots ignoriert werden muessen, ohne in Logik der Berechnungen zu gehen.

## Stile der Indikator in den Beispielen

MetaTrader 5 Client Terminal enthielt 38 eingebaute Technische Indikatoren, die in MQL5-Programmen mithilfe der [entsprechenden Funktionen](#) verwendet werden können. Aber der Hauptvorteil der MQL5-Sprache ist die Fähigkeit, eigene Indikatoren, die dann in Expert Advisor für Erhaltung der Werte oder für die technische Analyse verwendet werden, zu erstellen.

Das ganze Reihe von Indikatoren kann aus mehreren grundlegenden [Zeichenstile](#), die grafische graphische Konstruktionen genannt sind, abgeleitet werden. Eine Konstruktion ist eine Weise der Darstellung von Daten, die der Indikator berechnet, speichert und zeigt auf Aufruf. Es gibt sieben solcher grundlegenden Konstruktionen:

1. Linie
2. Sektion (Segment)
3. Histogramm
4. Pfeil (Symbol)
5. Die gefärbte Fläche (Kanal zu füllen)
6. Balken
7. Japanische Kerzen

Jede Darstellung benötigt 1-5 [Arrays](#) des Typs [double](#), in deren Indikatorwerte gespeichert werden. Für die Bequemlichkeit werden diese Arrays mit den Indikator-Puffern verbunden. Die Anzahl der Puffer im Indikator muss im Voraus durch [Compiler-Direktiven](#) deklariert werden, zum Beispiel:

```
#property indicator_buffers 3 // Anzahl der Puffer
#property indicator_plots 2 // Anzahl der graphischen Konstruktionen
```

Die Anzahl der Puffer im Indikator ist immer größer oder gleich der Anzahl von Konstrukten im Indikator.

Da jede grundlegende grafische Konstruktion kann Farbvariationen oder spezielle Darstellung haben, dann die tatsächliche Anzahl der Konstruktionen in der MQL5-Sprache ist 18:

| Konstruktion                 | Beschreibung                                                                                                             | Werte-Puffer | Farbpuffer |
|------------------------------|--------------------------------------------------------------------------------------------------------------------------|--------------|------------|
| <a href="#">DRAW_NONE</a>    | Wird nicht auf dem Chart visuell dargestellt, aber man kann die Werte des entsprechenden Puffers in "Data Window" finden | 1            | -          |
| <a href="#">DRAW_LINE</a>    | Eine Linie wird auf die Werte des entsprechenden Puffers gezeichnet (leere Werte im Puffer sind unerwünscht)             | 1            | -          |
| <a href="#">DRAW_SECTION</a> | Liniensegmente                                                                                                           | 1            | -          |

|                                        |                                                                                                                                      |   |   |
|----------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|---|---|
|                                        | werden zwischen den Werten des jeweiligen Puffers (in der Regel eine Menge von leeren Werten) gezeichnet                             |   |   |
| <a href="#"><u>DRAW_HISTOGRAM</u></a>  | Ein Histogramm wird von der Null-Linie bis den Werte des jeweiligen Puffer gezeichnet (leere Werte sind erlaubt)                     | 1 | - |
| <a href="#"><u>DRAW_HISTOGRAM2</u></a> | Ein Histogramm wird auf zwei Indikator-Puffer gezeichnet (leere Werte sind erlaubt)                                                  | 2 | - |
| <a href="#"><u>DRAW_ARROW</u></a>      | Wird als Symbols gezeichnet (leere Werte sind erlaubt)                                                                               | 1 | - |
| <a href="#"><u>DRAW_ZIGZAG</u></a>     | Ist dem Stil <a href="#"><u>DRAW_SECTION</u></a> ähnlich, aber im Gegensatz zu ihm kann er vertikale Segmente auf einem Balken bauen | 2 | - |
| <a href="#"><u>DRAW_FILLING</u></a>    | Farbfüllung zwischen zwei Linien. In "Data Window" werden zwei Werte der entsprechenden Puffer angezeigt                             | 2 | - |
| <a href="#"><u>DRAW_BARS</u></a>       | Wird als Balken auf dem Chart gezeichnet. In "Data Window" werden vier Werte der entsprechenden Puffer angezeigt                     | 4 | - |
| <a href="#"><u>DRAW_CANDLES</u></a>    | Wird als Kerzen auf dem Chart gezeichnet. In "Data Window" werden vier Werte der entsprechenden Puffer angezeigt                     | 4 | - |



|                                              |                                                                                                                                                                      |   |   |
|----------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|---|
| <a href="#"><u>DRAW_COLOR_LINE</u></a>       | Eine Linie, für die Sie Farben auf verschiedenen Bars ändern sowie ihre Farbe jederzeit wechseln können                                                              | 1 | 1 |
| <a href="#"><u>DRAW_COLOR_SECTION</u></a>    | Ist dem Stil <a href="#"><u>DRAW_SECTION</u></a> ähnlich, aber die Farbe jeder Sektion kann individuell eingegeben werden; die Farbe kann dynamisch angegeben werden | 1 | 1 |
| <a href="#"><u>DRAW_COLOR_HISTOGRAM</u></a>  | Ist dem Stil <a href="#"><u>DRAW_HISTOGRAM</u></a> ähnlich, aber jede Band kann ihre eigene Farbe haben; die Farbe kann dynamisch angegeben werden                   | 1 | 1 |
| <a href="#"><u>DRAW_COLOR_HISTOGRAM2</u></a> | Ist dem Stil <a href="#"><u>DRAW_HISTOGRAM2</u></a> ähnlich, aber jede Band kann ihre eigene Farbe haben; die Farbe kann dynamisch angegeben werden                  | 2 | 1 |
| <a href="#"><u>DRAW_COLOR_ARROW</u></a>      | Ist dem Stil <a href="#"><u>DRAW_ARROW</u></a> ähnlich, aber jedes Symbol kann seine eigene Farbe haben. Farbe kann dynamisch geändert werden                        | 1 | 1 |
| <a href="#"><u>DRAW_COLOR_ZIGZAG</u></a>     | Stil <a href="#"><u>DRAW_ZIGZAG</u></a> mit den Fähigkeiten der individuellen Farbgebung der Sektionen und dynamischen Farbwechsel                                   | 2 | 1 |
| <a href="#"><u>DRAW_COLOR_BARS</u></a>       | Stil <a href="#"><u>DRAW_BARS</u></a> mit den Fähigkeiten der individuellen Farbgebung der                                                                           | 4 | 1 |

|                                    |                                                                                                                           |   |   |
|------------------------------------|---------------------------------------------------------------------------------------------------------------------------|---|---|
|                                    | Balken und dynamischen Farbwechsel                                                                                        |   |   |
| <a href="#">DRAW_COLOR_CANDLES</a> | Stil <a href="#">DRAW_CANDLES</a> mit den Fähigkeiten der individuellen Farbgebung der Kerzen und dynamischen Farbwechsel | 4 | 1 |

## Der Unterschied zwischen einem Indikator-Puffer und einem Array

In jedem Indikator deklarieren wir ein oder mehr Arrays vom Typ `double` auf die [globale Ebene](#), die dann als Indikator-Puffer mit der [SetIndexBuffer\(\)](#)-Funktion verwendet werden. Für Zeichnung der graphischen Konstruktionen eines Indiktors sind nur die Werte der Indikator-Puffer verwendet, alle anderen Arrays können nicht dafür genutzt werden. Darüber hinaus sind die Werte der Puffer in "Data Window" angezeigt.

Der Indikator-Puffer sollte [dynamisch](#) sein und erfordert keine [Angabe der Größe](#) - die Größe des Arrays, das als Indikator-Puffer verwendet wird, wird automatisch durch die Laufzeit-Subsystem des Terminals angegeben.

[Indizierung-Richtung](#) nach der Bindung des Arrays mit dem Indikator-Puffer ist standardmäßig wie in einem gewöhnlichen Array aufgestellt, aber Sie können die Funktion [ArraySetAsSeries\(\)](#) verwenden, um die Methode für den Zugriff zu Elemente eines Arrays zu ändern. Standardmäßig wird der Indikator-Puffer verwendet, um Daten für die Darstellung zu speichern ([INDICATOR\\_DATA](#)).

Wenn Berechnung der Indikator-Werte Zwischen-Berechnungen und Speicherung von untergeordneter Wert für jedem Balken erfordert, so können Sie dieses Array während Bindung als Berechnung-Puffer deklarieren ([INDICATOR\\_CALCULATIONS](#)). Für die Zwischen-Werte kann ein regelmäßiges Array verwendet werden, aber in diesem Fall muss der Programmierer selbst die Größe des Arrays steuern.

Einige Darstellungen erlauben Farbe für jeden Balken anzugeben. Um Informationen über die Farbe zu speichern, werden Farbpuffer verwendet ([INDICATOR\\_COLOR\\_INDEX](#)). Die Farbe ist ein Integer-Typ [color](#), aber nicht alle Indikator-Puffer müssen vom Typ [double](#) sein. Werte von der Farbpuffer und Hilfspuffer ([INDICATOR\\_CALCULATIONS](#)) können nicht mit der [CopyBuffer\(\)](#) function erhalten werden.

Die Anzahl der Indikator Puffer muss durch den Compiler-Direktive `#property indicator_buffers` Anzahl\_der\_Puffers angegeben werden:

```
#property indicator_buffers 3 // Der Indikator hat 3 Puffer
```

Die maximale Anzahl der Puffer in einem Indikator ist 512.

## Relevanz von Indikator-Puffer und grafische Konstruktionen

Jede grafische Konstruktion basiert auf einem oder mehreren Indikator-Puffer. Zum Beispiel, Kerze

brauchen vier Werte - die Preise von Open, High, Low und Close. Dementsprechend müssen Sie für den Indikator in Form von Kerzen 4 Indikator-Puffer und 4 Arrays vom Typ double für sie deklarieren. Zum Beispiel:

```
//--- Ein Indikator hat vier Puffer
#property indicator_buffers 4
//--- Der Indikator hat eine graphische Konstruktion
#property indicator_plots 1
//--- Graphische Konstruktion Nummer 1 wird als Kerzen gezeichnet
#property indicator_type1 DRAW_CANDLES
//--- Die Farbe der Kerzen ist clrDodgerBlue
#property indicator_color1 clrDodgerBlue
//--- 4 Arrays für die Puffer
double OBuffer[];
double HBuffer[];
double LBuffer[];
double CBuffer[];
```

Grafische Darstellungen verwenden automatisch die Indikator-Puffer in Übereinstimmung mit der Darstellung-Nummer. Nummer der Darstellungen beginnen von eins, Nummer der Puffer beginnen von Null. Wenn die erste Darstellung 4 Indikator-Puffer benötigt, dann werden erste 4 Indikator-Puffer für Zeichnung verwendet. Diese vier Puffer sollten durch die Funktion [SetIndexBuffer\(\)](#) mit den entsprechenden Arrays mit der richtigen Indizierung gebunden werden.

```
//--- Bindung von Arrays und Indikator-Puffer
SetIndexBuffer(0,OBuffer,INDICATOR_DATA); // Der erste Puffer entspricht den Index
SetIndexBuffer(1,HBuffer,INDICATOR_DATA); // Der zweite Puffer entspricht den Index
SetIndexBuffer(2,LBuffer,INDICATOR_DATA); // Der dritte Puffer entspricht den Index
SetIndexBuffer(3,CBuffer,INDICATOR_DATA); // Der vierte Puffer entspricht den Index
```

Beim Zeichnung von Kerzen, verwendet der Indikator die erste vier Puffer, da sie Darstellung "Kerze" von der ersten Reihe deklariert wurde.

Wir ändern das Beispiel und fügen eine Konstruktion von einer einfacher Linie hinzu - [DRAW\\_LINE](#). Nehmen wir nun an, dass die Linie Nummer 1 hat und die Kerzen Nummer 2 haben. Die Anzahl der Puffer und die Anzahl der Konstruktionen haben erhöht.

```
//--- Der Indikator hat fünf Indikator-Puffer
#property indicator_buffers 5
//--- Der Indikator hat zwei graphischen Konstruktionen
#property indicator_plots 2
//--- Graphische Konstruktion Nummer 1 wird als eine Linie gezeichnet
#property indicator_type1 DRAW_LINE
//--- Die Farbe der Linie ist clrDodgerRed
#property indicator_color1 clrDodgerRed
//--- Graphische Konstruktion Nummer 2 wird als Kerzen gezeichnet
#property indicator_type2 DRAW_CANDLES
//--- Die Farbe der Kerzen ist clrDodgerBlue
#property indicator_color2 clrDodgerBlue
```

```
//--- 5 Arrays für die Puffer
double LineBuffer[];
double OBuffer[];
double HBuffer[];
double LBuffer[];
double CBuffer[];
```

Die Reihenfolge der Konstruktionen hat verändert, jetzt kommt eine Linie und dann Kerzen. Daher wird die Reihenfolge der Puffer die gleiche sein - zum ersten Mal deklarieren wir den Linie-Puffer für Index Null, und dann vier Puffer für Kerzen.

```
SetIndexBuffer(0,LineBuffer,INDICATOR_DATA); // Der erste Puffer entspricht den I
//--- Bindung von Arrays und Indikator-Puffer für Kerzen
SetIndexBuffer(1,OBuffer,INDICATOR_DATA); // Der zweite Puffer entspricht den I
SetIndexBuffer(2,HBuffer,INDICATOR_DATA); // Der dritte Puffer entspricht den I
SetIndexBuffer(3,LBuffer,INDICATOR_DATA); // Der vierte Puffer entspricht den I
SetIndexBuffer(4,CBuffer,INDICATOR_DATA); // Der fünfte Puffer entspricht den I
```

Die Anzahl der Puffer und grafischen Konstruktionen können nur mit Compiler-Direktiven angegeben werden, die dynamischen Veränderungen dieser Eigenschaften mit Funktionen ist unmöglich.

## Farbversionen von Stilen

Stile sind in zwei Gruppen eingeteilt. Die erste Gruppe enthält die Stile, in denen es kein Wort **COLOR** im Titel gibt, wir nennen diese Stile grundlegende Stile:

- DRAW\_LINE
- DRAW\_SECTION
- DRAW\_HISTOGRAM
- DRAW\_HISTOGRAM2
- DRAW\_ARROW
- DRAW\_ZIGZAG
- DRAW\_FILLING
- DRAW\_BARS
- DRAW\_CANDLES

Die zweite Gruppe enthält die Stile mit dem Wort **COLOR** in Titel, wir nennen sie Farbversionen:

- DRAW\_COLOR\_LINE
- DRAW\_COLOR\_SECTION
- DRAW\_COLOR\_HISTOGRAM
- DRAW\_COLOR\_HISTOGRAM2
- DRAW\_COLOR\_ARROW
- DRAW\_COLOR\_ZIGZAG
- DRAW\_COLOR\_BARS
- DRAW\_COLOR\_CANDLES

Alle Farbversionen der Stile unterscheiden sich von den grundlegenden Stile in dass sie eine Farbe für jeden Teil der graphischen Konstruktion anzugeben erlauben. Minimaler Teil der Konstruktion ist ein Balken, so kann man sagen, dass die Farbversionen die Farbe der Konstruktion auf jedem Balken einzugeben erlauben.

Um die Farbe der Konstruktion auf jeden Balken anzugeben, erhalten die Farbversionen der Stile ein zusätzlicher spezieller Puffer, um die Farbindizes zu speichern. Diese Indizes zeigen die Nummer der Farbe in einem speziellen Array mit einem vordefinierten Set von Farben. Die Größe des Arrays von Farben ist 64. Dies bedeutet, dass jede Farbversion erlaubt Färbung mit 54 verschiedenen Farben.

Das Set und die Anzahl der Farben im speziellen Array kann dynamisch durch Compiler-Direktive `#property indicator_color` angegeben werden (durch Kommata werden alle notwendigen Farben angegeben). Zum Beispiel, der Eintrag im Indikator:

```
//--- Wir definieren 8 Farben zum Einfärben von Kerzen (sie sind in dem speziellen Array)
#property indicator_color1 clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrLightBlue
```

Hier ist angegeben, dass für die grafische Konstruktion Nummer acht Farben, die in einem speziellen Array gestellt werden, definiert sind. Als nächstes werden wir nicht die Farbe angeben, sondern nur ihr Index. Wenn wir rot für den Balken angeben sollen, dann müssen wir den Index der roten Farbe aus einem Array in Farbpuffer angeben. Die rote Farbe hat Nummer 1 in Direktive, Indexnummer 0 entspricht sie.

```
//--- Definieren wir die Farbe der Kerze clrRed
col_buffer[buffer_index]=0;
```

Ein Set von Farben für die Färbung kann dynamisch durch `PlotIndexSetInteger()`-Funktion verändert sein. Beispiel:

```
//--- Wir geben die Farbe für jeden Index als die Eigenschaft PLOT_LINE_COLOR
PlotIndexSetInteger(0, // Nummer der Grafikstile
    PLOT_LINE_COLOR, // Eigenschaftbezeichnung
    plot_color_ind, // Index der Farbe, in dem wir die Farbe ändern
    color_array[i]); // Neue Farbe
```

## Eigenschaften des Indikators und der grafischen Konstruktionen

Eigenschaften für graphischen Konstruktionen können durch [Compiler-Direktiven](#) und mit den entsprechenden Funktionen angegeben werden. Lesen Sie mehr darüber in der Sektion [Zusammenhang zwischen Eigenschaften des Indikators und entsprechenden Funktionen](#). Dynamische Veränderung der Indikator-Eigenschaften durch Funktionen erlaubt es flexibler benutzerdefinierte Indikatoren zu erstellen.

## Start der Zeichnung des Indikators im Chart

In vielen Fällen, unter den Bedingungen des Algorithmus, kann die Berechnung der Indikatorwerte nicht sofort mit dem aktuellen Balken gestartet werden, es ist erforderlich, die minimale Anzahl von vorgehenden Balken in der Geschichte zu versorgen. Zum Beispiel bedeuten viele Arten von Glättung, dass ein Array von Preisen für das vorgehenden N Balken erforderlich ist, und auf der Grundlage dieser

Werte wird der Indikatorwert für den aktuellen Balken berechnet.

In solchen Fällen entweder gibt es keine Möglichkeit, die Indikatorwerte für die ersten N Balken zu berechnen, oder diese Werte sollen nicht auf dem Chart angezeigt werden und sind nur subsidiär für die Berechnung der folgenden Werte. Um nichts auf die erste N Balken in der Geschichte zu zeichnen, geben Sie den N Wert der Eigenschaft [PLOT\\_DRAW\\_BEGIN](#) für die entsprechende grafische Konstruktion ein:

```
//--- Bindung von Arrays und Indikator-Puffer für Kerzen  
PlotIndexSetInteger(Nummer_der_graphischen_Konstruktion, PLOT_DRAW_BEGIN, N);
```

Hier:

- Nummer\_der\_graphischen\_Konstruktion - Wert von 0 bis indicator\_plots-1 (Nummerierung der Konstruktionen beginnt mit Null).
- N - Anzahl der ersten Balken in der Geschichte, auf denen der Indikator nicht auf dem Chart angezeigt werden soll.

## DRAW\_NONE

Stil DRAW\_NONE ist für diejenigen Fälle, in denen der Puffer-Wert berechnet und in "Data Window" dargestellt werden sollte, aber die Anzeige auf dem Chart ist nicht erforderlich. Um die Genauigkeit der Anzeige zu konfigurieren, verwenden Sie Ausdruck `IndicatorSetInteger(INDICATOR_DIGITS,Anzahl_der_Zeichen)` in der `OnInit()`-Funktion:

```
int OnInit()
{
    //--- indicator buffers mapping
    SetIndexBuffer(0,InvisibleBuffer,INDICATOR_DATA);
    //--- Wir stellen die Genauigkeit, mit der der Wert im Daten-Fenster angezeigt werden
    IndicatorSetInteger(INDICATOR_DIGITS,0);
    //---
    return(INIT_SUCCEEDED);
};
```

Die erforderliche Anzahl der Puffer für den Bau von DRAW\_NONE ist 1.

Ein Beispiel für einen Indikator, der in "Data Window" die Nummer des Balkens, auf dem die Maus ist, zeigt. Die Nummerierung entspricht der Zeit-Serie, d.h. der aktuelle unvollendete Balken hat Null-Index, und der älteste Balken verfügt über den größten Index.



Beachten Sie, dass trotz der Tatsache, dass für die grafische Konstruktion **Nummer 1** rote Farbe der Darstellung angegeben ist, zeichnet der Indikator nichts auf dem Chart.

```
//+-----+
//|                                     DRAW_NONE.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
```

```

//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots  1
//--- plot Invisible
#property indicator_label1  "Bar Index"
#property indicator_type1   DRAW_NONE
#property indicator_style1  STYLE_SOLID
#property indicator_color1  clrRed
#property indicator_width1  1
//--- indicator buffers
double      InvisibleBuffer[];
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- Bindung vom Array und Indikator-Puffer
    SetIndexBuffer(0,InvisibleBuffer,INDICATOR_DATA);
//--- Wir stellen die Genauigkeit, mit der der Wert in Data Window angezeigt werden
    IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
    return(INIT_SUCCEEDED);
};
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
    static datetime lastbar=0;
//--- Wenn dies die erste Berechnung des Indikators ist
    if(prev_calculated==0)
    {
        //--- Durchnummerieren Balken zum ersten Mal
        CalcValues(rates_total,close);
        //--- Speichern Öffnungszeit des aktuellen Balkens in lastbar
        lastbar=(datetime)SeriesInfoInteger(_Symbol,_Period,SERIES_LASTBAR_DATE);
    }
}

```



```

    ;}
else
{
    //--- Wenn ein neuer Balken erscheint hat, sien Öffnungszeit fällt nicht mit las
    if(lastbar!=SeriesInfoInteger(_Symbol,_Period,SERIES_LASTBAR_DATE))
    {
        //--- Durchnummerieren Balken noch einmal
        CalcValues(rates_total,close);
        //--- Aktualisieren Öffnungszeit des aktuellen Balkens in lastbar
        lastbar=(datetime)SeriesInfoInteger(_Symbol,_Period,SERIES_LASTBAR_DATE);
    }
    ;}

//--- return value of prev_calculated for next call
return(rates_total);
;}

//+-----+
//| Numeriert Balken wie in Zeitreihen |
//+-----+
void CalcValues(int total,double const &array[])
{
    //--- Wir definieren Indexierung des Indikator-Puffers als in Zeitreihen
    ArraySetAsSeries(InvisibleBuffer,true);
    //--- Wir füllen Nummer jedem Balken
    for(int i=0;i<total;i++) InvisibleBuffer[i]=i;
    ;}

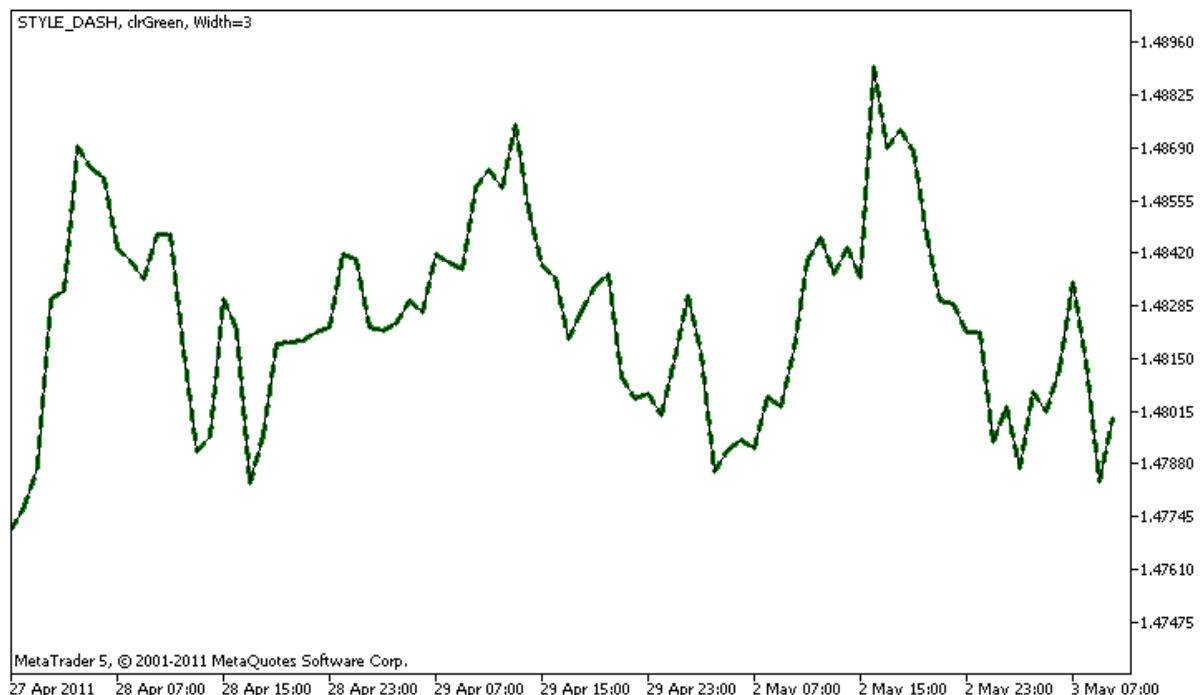
```

## DRAW\_LINE

Stil DRAW\_LINE zeichnet eine Linie der angegebenen Farbe auf die Werte des Indikator-Puffers. Die Breite, Stil und Farbe der Linie können durch [Compiler-Direktiven](#), oder dynamisch mit der [PlotIndexSetInteger\(\)](#)-Funktion angegeben werden. Dynamische Veränderungen in den Eigenschaften der graphischen Konstruktion ermöglichen es Ihnen, die Indikatoren zu "wiederbeleben" so, dass sie ihr Aussehen ändern je nach der aktuellen Situation.

Die erforderliche Anzahl der Puffer für den Bau von DRAW\_LINE ist 1.

Ein Beispiel für einen Indikator, der eine Linie auf die Close Preise der Balken zeichnet. Farbe, Breite und Linienart werden nach dem Zufallsprinzip jede N=5 Ticks geändert.



Bitte beachten Sie, dass für plot1 mit dem DRAW\_LINE-Stil werden Eigenschaften mithilfe Compiler-Direktive [#property](#) angegeben, und dann in der Funktion [OnCalculate\(\)](#) werden diese drei Eigenschaften nach dem Zufallsprinzip angegeben. Der Parameter N wird in [externen Parametern](#) des Indikators angegeben, damit kann es manuell angegeben werden (die Registerkarte "Einstellungen" im Indikatoreigenschaften-Fenster).

```
//+-----+
//|                                     DRAW_LINE.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Der Indikator demonstriert DRAW_LINE"
#property description "Zeichnet eine Linie der angegebenen Farbe auf die Close Preise"
#property description "Farbe, Breite und Linienart werden nach dem Zufallsprinzip"
```

```

#property description "jede N Ticks verändert"

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots 1
//--- Eigenschaften der Linie werden durch Compiler-Direktiven definiert
#property indicator_label1 "Line" // Name der Darstellung für Data Window
#property indicator_type1 DRAW_LINE // Typ der graphischen Darstellung ist die Linie
#property indicator_color1 clrRed // Linienfarbe
#property indicator_style1 STYLE_SOLID // Linienstil
#property indicator_width1 1 // Linienbreite
//--- Eingabeparameter
input int N=5; // Anzahl der Ticks für Änderung
//--- Indikator-Puffer
double LineBuffer[];
//--- Array, um die Farbe zu speichern
color colors[]={clrRed,clrBlue,clrGreen};
//--- Array für Speicherung der Linienstile
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOTDOT};
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- Bindung vom Array und Indikator-Puffer
SetIndexBuffer(0,LineBuffer,INDICATOR_DATA);
//--- Initialisierung des Zufallszahlengenerators
MathSrand(GetTickCount());
//---
return(INIT_SUCCEEDED);
};
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
    static int ticks=0;
//--- Zählen wir Ticks, um den Stil, die Farbe und Breite der Linie zu ändern
    ticks++;
//--- Wenn eine kritische Anzahl von Ticks angesammelt hat

```

```

    if(ticks>=N)
    {
        //--- Ändern wir die Eigenschaften der Linie
        ChangeLineAppearance();
        //--- Setzen wir den Zähler der Ticks auf Null
        ticks=0;
    };

//--- Block mit Berechnung der Indikatorwerte
for(int i=0;i<rates_total;i++)
{
    LineBuffer[i]=close[i];
};

//--- den Wert prev_calculated für den nächsten Anruf der Funktion zurückgeben
return(rates_total);
};

//+-----+
//| Ändert das Aussehen der angezeigten Linie im Indikator |
//+-----+
void ChangeLineAppearance()
{
    //--- String um Informationen über die Eigenschaften der Linie zu erstellen
    string comm="";
    //--- Block mit Änderungen der Linienfarbe
    //--- Wir erhalten eine Zufallszahl
    int number=MathRand();
    //--- Teiler der Zahl entspricht der Größe des Arrays colors[]
    int size=ArraySize(colors);
    //--- Erhalten wir den Index als den Rest der ganzzahligen Division, um eine neue Farbe
    int color_index=number%size;
    //--- Geben wir die Farbe als Eigenschaft PLOT_LINE_COLOR an
    PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
    //--- Schreiben wir die Linienfarbe
    comm=comm+(string)colors[color_index];

    //--- Block mit Änderungen der Linienbreite
    number=MathRand();
    //--- Erhalten wir die Breite als den Rest der ganzzahligen Division
    int width=number%5; // Breite ist von 0 bis 4
    //--- Geben wir die Farbe als Eigenschaft PLOT_LINE_WIDTH
    PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
    //--- Schreiben wir die Linienbreite
    comm=comm+", Width="+IntegerToString(width);

    //--- Block mit Änderungen des Linienstils
    number=MathRand();
    //--- Teiler der Zahl entspricht der Größe des Arrays styles
    size=ArraySize(styles);

```

```
//--- Erhalten wir den Index als den Rest der ganzzahligen Division, um ein neuer Stil
    int style_index=number%size;
//--- Geben wir die Farbe als Eigenschaft PLOT_LINE_COLOR an
    PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- Schreiben wir den Linienstil
    comm=EnumToString(styles[style_index])+", "+comm;
//--- Zeigen wir Informationen auf dem Chart durch den Kommentar
    Comment(comm);
; }
```



```

//|                                     DRAW_SECTION.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Der Indikator demonstriert DRAW_SECTION"
#property description "Zeichnet gerade Linien durch jede bars Balken"
#property description "Farbe, Breite und Stil der Sektionen werden zufällig"
#property description "jede N Ticks verändert"

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot Section
#property indicator_label1  "Section"
#property indicator_type1   DRAW_SECTION
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- Eingabeparameter
input int      bars=5;          // Länge der Sektionen in Balken
input int      N=5;             // Anzahl der Ticks um Stil der Sektionen zu ändern
//--- Indicator-Puffer
double         SectionBuffer[];
//--- Hilfsvariable, um Ende der Sektionen zu berechnen
int            divider;
//--- Array, um die Farbe zu speichern
color colors[]={clrRed,clrBlue,clrGreen};
//--- Array für Speicherung der Linienstile
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOTDOT}
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- Bindung vom Array und Indikator-Puffer
SetIndexBuffer(0,SectionBuffer,INDICATOR_DATA);
//--- Der Leerer Wert (0) wird nicht in der Zeichnung teilnehmen
PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- Überprüfen wir die Indikator-Parameter
if(bars<=0)
{
PrintFormat("Invalid Parameterwert bar=%d",bars);
return(INIT_PARAMETERS_INCORRECT);
};
else divider=2*bars;

```

```

//---+
    return(INIT_SUCCEEDED);
; }
//+-----+
//| Custom indicator iteration function |
//+-----+

int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
    static int ticks=0;
//--- Zählen wir Ticks, um den Stil, die Farbe und Breite der Linie zu ändern
    ticks++;
//--- Wenn eine kritische Anzahl von Ticks angesammelt hat
    if(ticks>=N)
    {
        //--- Ändern wir die Eigenschaften der Linie
        ChangeLineAppearance();
        //--- Setzen wir den Zähler der Ticks auf Null
        ticks=0;
    ; }

//--- Anzahl der Balken, mit dem die Berechnung des Indikators beginnt
    int start=0;
//--- Wenn der Indikator bevor berechnet wurde, dann setzen wir start bei dem vorherigen
    if(prev_calculated>0) start=prev_calculated-1;
//--- alle Berechnungen der Indikatorwerte
    for(int i=start;i<rates_total;i++)
    {
        //--- Wir bekommen Rest nach der Division der Nummer des Balkens durch 2*bars
        int rest=i%divider;
        //--- Wenn die Nummer des Balken ohne Rest durch 2*bars teilbar ist
        if(rest==0)
        {
            //--- Ende der Segment ist auf den High-Preis dieses Balkens gesetzt
            SectionBuffer[i]=high[i];
        ; }
        //--- Wenn der Rest der Division gleich bars ist,
        else
        {
            //--- Ende der Segment ist auf den High-Preis dieses Balkens gesetzt
            if(rest==bars) SectionBuffer[i]=low[i];

```



```

//--- Wenn etwas nicht passt, dann überspringen wir diesen Balken - setzen 0
    else SectionBuffer[i]=0;
    ;}

    ;}

//--- den Wert prev_calculated für den nächsten Anruf der Funktion zurückgeben
    return(rates_total);
    ;}

//+-----+
//| Ändert das Aussehen der Sektionen im Indikator |
//+-----+
void ChangeLineAppearance()
{
//--- String um Informationen über die Eigenschaften der Linie zu erstellen
    string comm="";
//--- Block mit Änderungen der Linienfarbe
    int number=MathRand(); // Erhalten wir Zufallszahl
//--- Teiler der Zahl entspricht der Größe des Arrays colors[]
    int size=ArraySize(colors);
//--- Erhalten wir den Index als den Rest der ganzzahligen Division, um eine neue Farbe
    int color_index=number%size;
//--- Geben wir die Farbe als Eigenschaft PLOT_LINE_COLOR an
    PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- Schreiben wir die Linienfarbe
    comm=comm+"\r\n"+(string)colors[color_index];

//--- Block mit Änderungen der Linienbreite
    number=MathRand();
//--- Erhalten wir die Breite als den Rest der ganzzahligen Division
    int width=number%5; // Breite ist von 0 bis 4
//--- Breite angeben
    PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Schreiben wir die Linienbreite
    comm=comm+"\r\nWidth="+IntegerToString(width);

//--- Block mit Änderungen des Linienstils
    number=MathRand();
//--- Teiler der Zahl entspricht der Größe des Arrays styles
    size=ArraySize(styles);
//--- Erhalten wir den Index als den Rest der ganzzahligen Division, um ein neuer Stil
    int style_index=number%size;
//--- Linienart angeben
    PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- Schreiben wir den Linienstil
    comm="\r\n"+EnumToString(styles[style_index])+" "+comm;
//--- Zeigen wir Informationen auf dem Chart durch den Kommentar
    Comment(comm);
    ;}

```

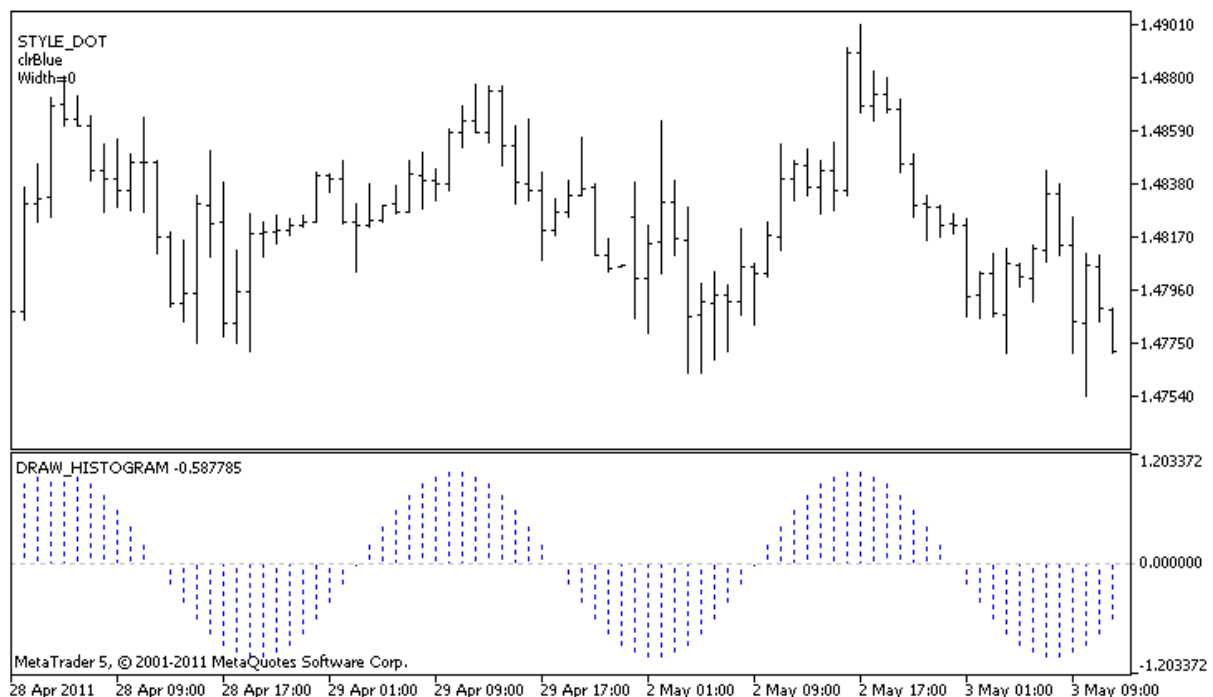
## DRAW\_HISTOGRAM

DRAW\_HISTOGRAM-Stil zeichnet ein Histogramm der angegebenen Farbe von Null bis den angegebenen Wert. Die Werte werden aus den Indikator-Puffer aufgenommen. Die Breite, Farbe und Stil der Säule können wie für [DRAW\\_LINE](#) durch [Compiler-Direktiven](#) oder dynamisch mit der [PlotIndexSetInteger\(\)](#)-Funktion angegeben werden. Dynamische Veränderungen in den Eigenschaften der graphischen Konstruktion ermöglichen es Ihnen, den Aussehen des Histogramms je nach der aktuellen Situation zu ändern.

Da wird auf jeden Balken eine Säule von Null-Ebene gezeichnet, dann wird DRAW\_HISTOGRAM besser für Zeichnung in einem separaten Fenster verwendet. Meistens wird diese Art der graphischen Konstruktion an einen Oszillator Art von Indikatoren, z.B. [Bears Power](#) oder [OsMA](#) verwendet. Für leere nicht gezeichnete Werte ist es genug Null-Werte anzugeben.

Die erforderliche Anzahl der Puffer für den Bau von DRAW\_HISTOGRAM ist 1.

Ein Beispiel für einen Indikator, der eine Sinuskurve der angegebenen Farbe auf Funktion [MathSin\(\)](#) zeichnet. Die Farbe, Breite und Stil **aller** Säule des Histogramms werden nach dem Zufallsprinzip jede **N** Ticks verändert. Parameter bars gibt den Zeitraum der Sinuskurve, d.h. in einer bestimmten Anzahl von Balken wird die Sinuskurve ihren Zyklus wiederholen.



Bitte beachten Sie, dass für **plot1** mit dem DRAW\_HISTOGRAM-Stil werden Eigenschaften mithilfe Compiler-Direktive [#property](#) angegeben, und dann in der Funktion [OnCalculate\(\)](#) werden diese drei Eigenschaften nach dem Zufallsprinzip angegeben. Der Parameter N wird in [externen Parametern](#) des Indikators angegeben, damit kann es manuell angegeben werden (die Registerkarte "Einstellungen" im Indikatoreigenschaften-Fenster).

```
//+-----+
//|                                     DRAW_HISTOGRAM.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
```

```
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Der Indikator demonstriert DRAW_HISTOGRAM"
#property description "Zeichnet die Sinuskurve als ein Histogramm in einem separaten Fenster"
#property description "Farbe und Breite der Säulen werden zufällig"
#property description "jede N Ticks verändert"
#property description "Parameter bars setzt die Anzahl der Balken im Zyklus der Sinuskurve"

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots 1
//--- plot Histogram
#property indicator_label1 "Histogram"
#property indicator_type1 DRAW_HISTOGRAM
#property indicator_color1 clrBlue
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//--- Eingabeparameter
input int bars=30; // Zeitraum der Sinuskurve in Balken
input int N=5; // Anzahl der Ticks um Histogramm zu ändern
//--- indicator buffers
double HistogramBuffer[];
//--- Faktor für 2Pi Winkel im Bogenmaß durch Multiplikation mit der Parameter bars
double multipliern;
//--- Array, um die Farbe zu speichern
color colors[]={clrRed,clrBlue,clrGreen};
//--- Array für Speicherung der Linienstile
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOTDOT}
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- indicator buffers mapping
SetIndexBuffer(0,HistogramBuffer,INDICATOR_DATA);
//--- Berechnen wir den Multiplikator
if(bars>1)multipliern=2.*M_PI/bars;
else
{
PrintFormat("Geben Sie den Wert von bars=%d größer als 1",bars);
//--- Vorzeitige Beendigung des Indikators
return(INIT_PARAMETERS_INCORRECT);
};
//---
return(INIT_SUCCEEDED);
};
```

```

//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
    static int ticks=0;
    //--- Zählen wir Ticks, um den Stil, die Farbe und Breite der Linie zu ändern
    ticks++;
    //--- Wenn eine kritische Anzahl von Ticks angesammelt hat
    if(ticks>=N)
    {
        //--- Ändern wir die Eigenschaften der Linie
        ChangeLineAppearance();
        //--- Setzen wir den Zähler der Ticks auf Null
        ticks=0;
    }

    //--- Berechnen der Indikatorwerte
    int start=0;
    //--- Wenn die Berechnung auf einem vorherigen Lauf von OnCalculate gemacht war
    if(prev_calculated>0) start=prev_calculated-1; // Setzen wir den Beginn der Berechnung
    //--- Füllen den Indikator-Puffer mit Werte
    for(int i=start;i<rates_total;i++)
    {
        HistogramBuffer[i]=sin(i*multiplier);
    }
    //--- den Wert prev_calculated für den nächsten Anruf der Funktion zurückgeben
    return(rates_total);
}

//+-----+
//| Ändert das Aussehen der Linien im Indikator |
//+-----+
void ChangeLineAppearance()
{
    //--- String um Informationen über die Eigenschaften der Linie zu erstellen
    string comm="";
    //--- Block mit Änderungen der Linienfarbe
    int number=MathRand(); // Erhalten wir Zufallszahl
    //--- Teiler der Zahl entspricht der Größe des Arrays colors[]
    int size=ArraySize(colors);

```

```

//--- Erhalten wir den Index als den Rest der ganzzahligen Division, um eine neue Farbe
    int color_index=number%size;
//--- Geben wir die Farbe als Eigenschaft PLOT_LINE_COLOR an
    PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- Schreiben wir die Linienfarbe
    comm=comm+"\r\n"+(string)colors[color_index];

//--- Block mit Änderungen der Linienbreite
    number=MathRand();
//--- Erhalten wir die Breite als den Rest der ganzzahligen Division
    int width=number%5;    // Breite ist von 0 bis 4
//--- Breite angeben
    PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Schreiben wir die Linienbreite
    comm=comm+"\r\nWidth="+IntegerToString(width);

//--- Block mit Änderungen des Linienstils
    number=MathRand();
//--- Teiler der Zahl entspricht der Größe des Arrays styles
    size=ArraySize(styles);
//--- Erhalten wir den Index als den Rest der ganzzahligen Division, um ein neuer Stil
    int style_index=number%size;
//--- Linienart angeben
    PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- Schreiben wir den Linienstil
    comm="\r\n"+EnumToString(styles[style_index])+" "+comm;
//--- Zeigen wir Informationen auf dem Chart durch den Kommentar
    Comment(comm);
; }

```

## DRAW\_HISTOGRAM2

DRAW\_HISTOGRAM2-Stil zieht ein Histogramm der angegebenen Farbe - die vertikalen Segmente auf die Werte der zwei Indikator-Puffer. Die Breite, Farbe und Stil der Segmenten können wie für [DRAW\\_LINE](#) durch [Compiler-Direktiven](#) oder dynamisch mit der [PlotIndexSetInteger\(\)](#)-Funktion angegeben werden. Dynamische Veränderungen in den Eigenschaften der graphischen Konstruktion ermöglichen es Ihnen, den Aussehens des Histogramms je nach der aktuellen Situation zu ändern.

DRAW\_HISTOGRAM2-Stil kann in einem separaten Grafikfenster und im Hauptfenster verwendet werden. Leere Werte werden nicht gezeichnet, alle Werte in Indikator-Puffern müssen explizit angegeben werden. Puffer werden nicht mit einem leeren Wert initialisiert.

Die erforderliche Anzahl der Puffer für den Bau von DRAW\_HISTOGRAM2 ist 2.

Ein Beispiel für einen Indikator, der zeichnet eine vertikale Säule der angegebenen Farbe und Breite zwischen Open und Close. Die Farbe, Breite und Stil **aller** Säule des Histogramms werden nach dem Zufallsprinzip jede N Ticks verändert. Beim Start des Indikators in der [OnInit\(\)](#)-Funktion wird die Nummer des Wochentags, für den das Histogramm nicht gezeichnet wird, zufällig eingegeben - invisible\_day. Dazu wird ein leerer Wert eingegeben [PLOT\\_EMPTY\\_VALUE=0](#):

```
//--- Leerer Wert setzen
PlotIndexSetDouble (Index_der_Zeichnung_DRAW_SECTION, PLOT_EMPTY_VALUE, 0);
```



Bitte beachten Sie, dass für `plot1` mit dem DRAW\_HISTOGRAM2-Stil werden Eigenschaften mithilfe Compiler-Direktive [#property](#) angegeben, und dann in der Funktion [OnCalculate\(\)](#) werden diese drei Eigenschaften nach dem Zufallsprinzip angegeben. Der Parameter N wird in [externen Parametern](#) des Indikators angegeben, damit kann es manuell angegeben werden (die Registerkarte "Einstellungen" im Indikatoreigenschaften-Fenster).

```
//+-----+
//|                                     DRAW_HISTOGRAM2.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
```

```

//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Der Indikator demonstriert DRAW_HISTOGRAM2"
#property description "Zeichnet ein Segment auf jedem Balken zwischen Open und Close"
#property description "Farbe, Breite und Stil werden zufällig"
#property description "jede N Ticks verändert"

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots 1
//--- plot Histogram_2
#property indicator_label1 "Histogram_2"
#property indicator_type1  DRAW_HISTOGRAM2
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1 1
//--- input parameters
input int      N=5;                // Anzahl der Ticks um Histogram zu ändern
//--- indicator buffers
double         Histogram_2Buffer1[];
double         Histogram_2Buffer2[];
//--- der Wochentag, für den der Indikator nicht gezeichnet wird
int invisible_day;
//--- Array, um die Farbe zu speichern
color colors[]={clrRed,clrBlue,clrGreen};
//--- Array für Speicherung der Linienstile
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOTDOT}
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- indicator buffers mapping
    SetIndexBuffer(0,Histogram_2Buffer1,INDICATOR_DATA);
    SetIndexBuffer(1,Histogram_2Buffer2,INDICATOR_DATA);
//--- Leerer Wert setzen
    PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- Erhalten eine Zufallszahl von 0 bis 5
    invisible_day=MathRand()%6;
//---
    return(INIT_SUCCEEDED);
};
//+-----+
//| Custom indicator iteration function |
//+-----+

```

```

int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
    static int ticks=0;
    //--- Zählen wir Ticks, um den Stil, die Farbe und Breite der Linie zu ändern
    ticks++;
    //--- Wenn eine kritische Anzahl von Ticks angesammelt hat
    if(ticks>=N)
    {
        //--- Ändern wir die Eigenschaften der Linie
        ChangeLineAppearance();
        //--- Setzen wir den Zähler der Ticks auf Null
        ticks=0;
    }

    //--- Berechnen der Indikatorwerte
    int start=0;
    //--- Um den Wochentag durch den Öffnungszeiten jedes Balkens
    MqlDateTime dt;
    //--- Wenn die Berechnung auf einem vorherigen Lauf von OnCalculate gemacht war
    if(prev_calculated>0) start=prev_calculated-1; // Setzen wir den Beginn der Berechnung
    //--- Füllen den Indikator-Puffer mit Werte
    for(int i=start;i<rates_total;i++)
    {
        TimeToStruct(time[i],dt);
        if(dt.day_of_week==invisible_day)
        {
            Histogram_2Buffer1[i]=0;
            Histogram_2Buffer2[i]=0;
        }
        else
        {
            Histogram_2Buffer1[i]=open[i];
            Histogram_2Buffer2[i]=close[i];
        }
    }

    //--- den Wert prev_calculated für den nächsten Anruf der Funktion zurückgeben
    return(rates_total);
}

//+-----+
//| Ändert das Aussehen der Linien im Indikator |

```



```
//+-----+
void ChangeLineAppearance()
{
//--- String um Informationen über die Eigenschaften der Linie zu erstellen
    string comm="";
//--- Block mit Änderungen der Linienfarbe
    int number=MathRand(); // Erhalten wir Zufallszahl
//--- Teiler der Zahl entspricht der Größe des Arrays colors[]
    int size=ArraySize(colors);
//--- Erhalten wir den Index als den Rest der ganzzahligen Division, um eine neue Farbe
    int color_index=number%size;
//--- Geben wir die Farbe als Eigenschaft PLOT_LINE_COLOR an
    PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- Schreiben wir die Linienfarbe
    comm=comm+"\r\n"+(string)colors[color_index];

//--- Block mit Änderungen der Linienbreite
    number=MathRand();
//--- Erhalten wir die Breite als den Rest der ganzzahligen Division
    int width=number%5; // Breite ist von 0 bis 4
//--- Linienbreite angeben
    PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Schreiben wir die Linienbreite
    comm=comm+"\r\nWidth="+IntegerToString(width);

//--- Block mit Änderungen des Linienstils
    number=MathRand();
//--- Teiler der Zahl entspricht der Größe des Arrays styles
    size=ArraySize(styles);
//--- Erhalten wir den Index als den Rest der ganzzahligen Division, um ein neuer Stil
    int style_index=number%size;
//--- Linienart angeben
    PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- Schreiben wir den Linienstil
    comm="\r\n"+EnumToString(styles[style_index])+" "+comm;
//--- Informationen über den Tag, der in den Berechnungen ignoriert wird, hinzufügen
    comm="\r\nUngezeichnetert Tag - "+EnumToString((ENUM_DAY_OF_WEEK)invisible_day)+comm;
//--- Zeigen wir Informationen auf dem Chart durch den Kommentar
    Comment(comm);
};
```

## DRAW\_ARROW

Der DRAW\_ARROW-Stil zeichnet Pfeile (Zeichen aus dem [Wingdings](#)-Satz) der angegebenen Farbe auf den Wert des Indikator-Puffers. Die Breite und die Farbe der Symbole können wie für [DRAW\\_LINE](#) durch [Compiler-Direktiven](#) oder dynamisch mit der [PlotIndexSetInteger\(\)](#)-Funktion angegeben werden. Dynamische Veränderungen in den Eigenschaften der graphischen Konstruktion ermöglichen es Ihnen, den Aussehens eines Indikators zu ändern je nach der aktuellen Situation.

Der Symbolcode wird mit der [PLOT\\_ARROW](#)-Eigenschaft eingegeben.

```
//--- Den Symbolcode aus Wingdings-Satz für PLOT_ARROW definieren
PlotIndexSetInteger(0,PLOT_ARROW,code);
```

Der Standardwert ist PLOT\_ARROW=159 (Kreis).

Jeder Pfeil ist eigentlich ein Zeichen, das die Höhe und den Ankerpunkt hat, und kann einige wichtige Information über den Chart (z. B. der Schlusskurs am Balken) bedecken. Daher können Sie optional den vertikalen Verschiebung in Pixel, die nicht auf der Skala des Charts abhängig ist, angeben. Bei dieser Anzahl von Pixeln werden die Pfeile visuell vertikal verschoben, obwohl der Wert des Indikators gleich bleiben wird:

```
//--- Definieren wir vertikale Verschiebung der Pfeile in Pixel
PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
```

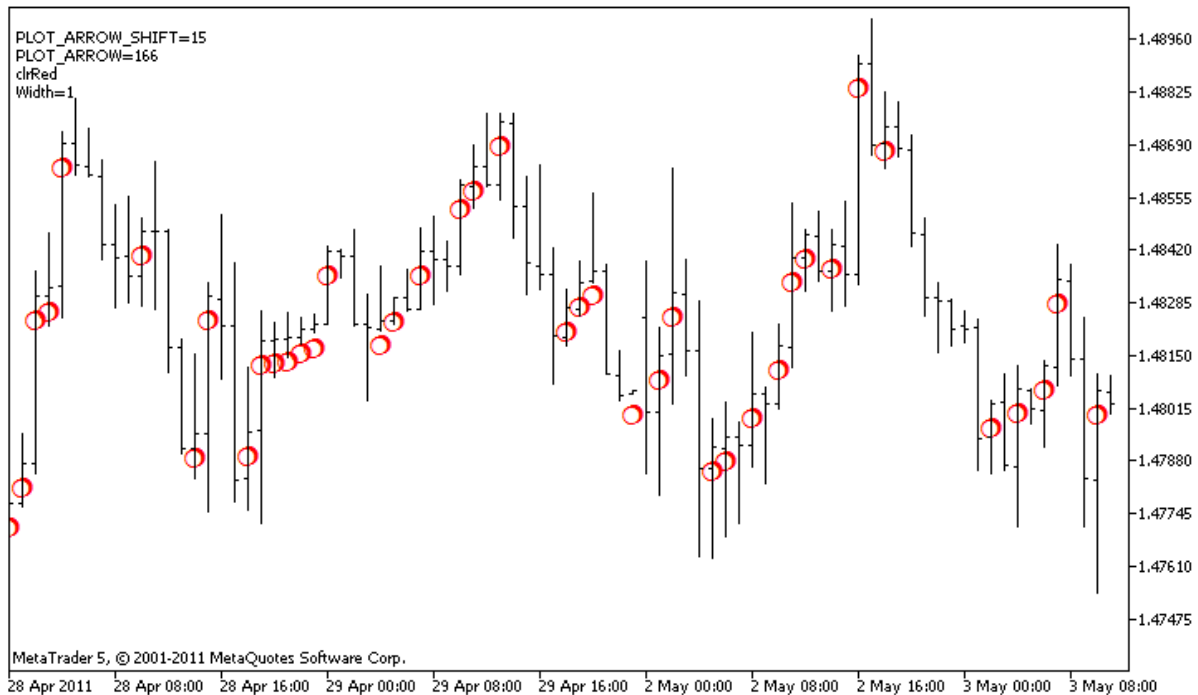
Ein negativer Wert von PLOT\_ARROW\_SHIFT bedeutet Verschiebung des Pfeils nach oben, ein positiver Wert verschiebt den Pfeil nach unten.

DRAW\_ARROW-Stil kann in einem separaten Grafikfenster und im Hauptfenster verwendet werden. Leere Werte werden nicht gezeichnet und nicht in den "Data Window" angezeigt, alle Werte in Indikator-Puffern müssen explizit angegeben werden. Puffer werden nicht mit einem leeren Wert initialisiert.

```
//--- Leerer Wert setzen
PlotIndexSetDouble(Index_der_Darstellung_DRAW_ARROW,PLOT_EMPTY_VALUE,0);
```

Die erforderliche Anzahl der Puffer für den Bau von DRAW\_ARROW ist 1.

Ein Beispiel des Indikators, der Pfeile auf jeder Balken, dessen Schlussprice Close mehr als Schlussprice des vorherigen Balkens ist, zeichnet. Die Farbe, Breite, Verschiebung und Zeichencode aller Pfeile werden nach dem Zufallsprinzip jede N Ticks verändert.



In diesem Beispiel, für `plot1` mit dem `DRAW_ARROW`-Stil werden Farbe und Größe mithilfe Compiler-Direktive `#Property` angegeben, und dann in der Funktion `OnCalculate()` werden Eigenschaften nach dem Zufallsprinzip angegeben. Der Parameter `N` wird in [externen Parametern](#) des Indikators angegeben, damit kann es manuell angegeben werden (die Registerkarte "Einstellungen" im Indikatoreigenschaften-Fenster).

```
//+-----+
//|                                     DRAW_ARROW.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Indikator zeigt DRAW_ARROW"
#property description "Auf dem Chart zeichnet es Pfeile, die durch Unicode-Symbole dargestellt werden"
#property description "Die Farbe, Größe, Verschiebung und Zeichencode des Pfeils werden zufällig gewählt"
#property description "jede N Ticks verändert"
#property description "Parameter code gibt den Basiswert an: Code=159 (Kreis)"

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots 1
//--- plot Arrows
#property indicator_label1  "Arrows"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrGreen
#property indicator_width1  1
//--- Eingabeparameter
```

```

input int      N=5;          // Anzahl der Ticks für Änderung
input ushort   code=159;     // Zeichencode zu zeichnen in DRAW_ARROW
//--- Indicator-Puffer
double         ArrowsBuffer[];
//--- Array, um die Farbe zu speichern
color colors[]={clrRed,clrBlue,clrGreen};
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- indicator buffers mapping
    SetIndexBuffer(0,ArrowsBuffer,INDICATOR_DATA);
//--- Geben wir Zeichencode um in PLOT_ARROW zu zeichnen an
    PlotIndexSetInteger(0,PLOT_ARROW,code);
//--- Definieren wir vertikale Verschiebung der Pfeile in Pixel
    PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,5);
//--- Setzen wir 0 als ein leerer Wert
    PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
    return(INIT_SUCCEEDED);
;}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{
    static int ticks=0;
//--- Zählen wir Ticks, um die Farbe, Größe, Verschiebung und Code des Pfeils zu ändern
    ticks++;
//--- Wenn eine kritische Anzahl von Ticks angesammelt hat
    if(ticks>=N)
    {
        //--- Ändern wir die Eigenschaften der Linie
        ChangeLineAppearance();
        //--- Setzen wir den Zähler der Ticks auf Null
        ticks=0;
    }
;}

//--- Block mit Berechnung der Indikatorwerte

```

```

    int start=1;
    if(prev_calculated>0) start=prev_calculated-1;
//--- Berechnungszyklus
    for(int i=1;i<rates_total;i++)
    {
        //--- Wenn der aktuelle Close-Preis ist höher als der vorherige Close-price, dann set
        if(close[i]>close[i-1])
            ArrowsBuffer[i]=close[i];
        //---Ansonsten geben einen leeren Wert an
        else
            ArrowsBuffer[i]=0;
    }
//--- return value of prev_calculated for next call
    return(rates_total);
};
//+-----+
//| Ändert das Aussehen der Zeichen im Indikator |
//+-----+
void ChangeLineAppearance()
{
    //--- String um Informationen über die Eigenschaften des Indikators zu erstellen
    string comm="";
    //--- Block mit Änderungen der Pfeilfarbe
    int number=MathRand(); // Erhalten wir Zufallszahl
    //--- Teiler der Zahl entspricht der Größe des Arrays colors[]
    int size=ArraySize(colors);
    //--- Erhalten wir den Index als den Rest der ganzzahligen Division, um eine neue Farbe
    int color_index=number%size;
    //--- Geben wir die Farbe als Eigenschaft PLOT_LINE_COLOR an
    PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
    //--- Schreiben wir die Linienfarbe
    comm=comm+"\r\n"+(string)colors[color_index];

    //--- Block mit Änderungen der Pfeilgröße
    number=MathRand();
    //--- Erhalten wir die Breite als den Rest der ganzzahligen Division
    int width=number%5; // die Größe wird von 0 bis 4 angegeben
    //--- Geben wir die Farbe als Eigenschaft PLOT_LINE_WIDTH
    PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
    //--- Schreiben wir die Größe der Pfeile
    comm=comm+"\r\nWidth="+IntegerToString(width);

    //--- Block mit Änderungen von Pfeilcode (PLOT_ARROW)
    number=MathRand();
    //--- Erhalten wir den Rest aus Integer-Division, um den neuen Pfeilcode zu berechnen
    int code_add=number%20;
    //--- Setzen wir einen neuen Zeichencode als die Summe von code+code_add
    PlotIndexSetInteger(0,PLOT_ARROW,code+code_add);
    //--- Schreiben wir den Zeichencode PLOT_ARROW

```

```
comm="\r\n"+"PLOT_ARROW="+IntegerToString(code+code_add)+comm;

//--- Block mit vertikaler Verschiebung der Pfeile in Pixel
number=MathRand();
//--- Erhalten wir die Verschiebung als den Rest der ganzzahligen Division
int shift=20-number%41;
//--- Einstellen wir eine neue Verschiebung von -20 bis 20
PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
//--- Schreiben wir die Verschiebung PLOT_ARROW_SHIFT
comm="\r\n"+"PLOT_ARROW_SHIFT="+IntegerToString(shift)+comm;

//--- Zeigen wir Informationen auf dem Chart durch den Kommentar
Comment(comm);

; }
```

## DRAW\_ZIGZAG

Stil DRAW\_ZIGZAG zeichnet Segmente der angegebenen Farbe auf die Werte der zwei Indikator-Puffer. Dieser Stil ist dem Stil [DRAW\\_SECTION](#) sehr ähnlich, aber im Gegensatz zu diesen können Sie die vertikale Segmente innerhalb eines Balkens zeichnen, wenn für diesen Balken Werte für die beiden Indikator-Puffer angegeben sind. Die Segmente werden vom dem Wert im ersten Puffer bis dem Wert im zweiten Indikator-Puffer gezeichnet. Keiner der Puffer darf nicht nur leere Werte erhalten, da in diesem Fall nichts gezeichnet wird.

Die Breite, Stil und Farbe der Linie können ebenso wie für den Stil [DRAW\\_SECTION](#) - durch [Compiler-Direktiven](#) oder dynamisch mit der [PlotIndexSetInteger\(\)](#)-Funktion angegeben werden. Dynamische Veränderungen in den Eigenschaften der graphischen Konstruktion ermöglichen es Ihnen, die Indikatoren zu "wiederbeleben" so, dass sie ihr Aussehen ändern je nach der aktuellen Situation.

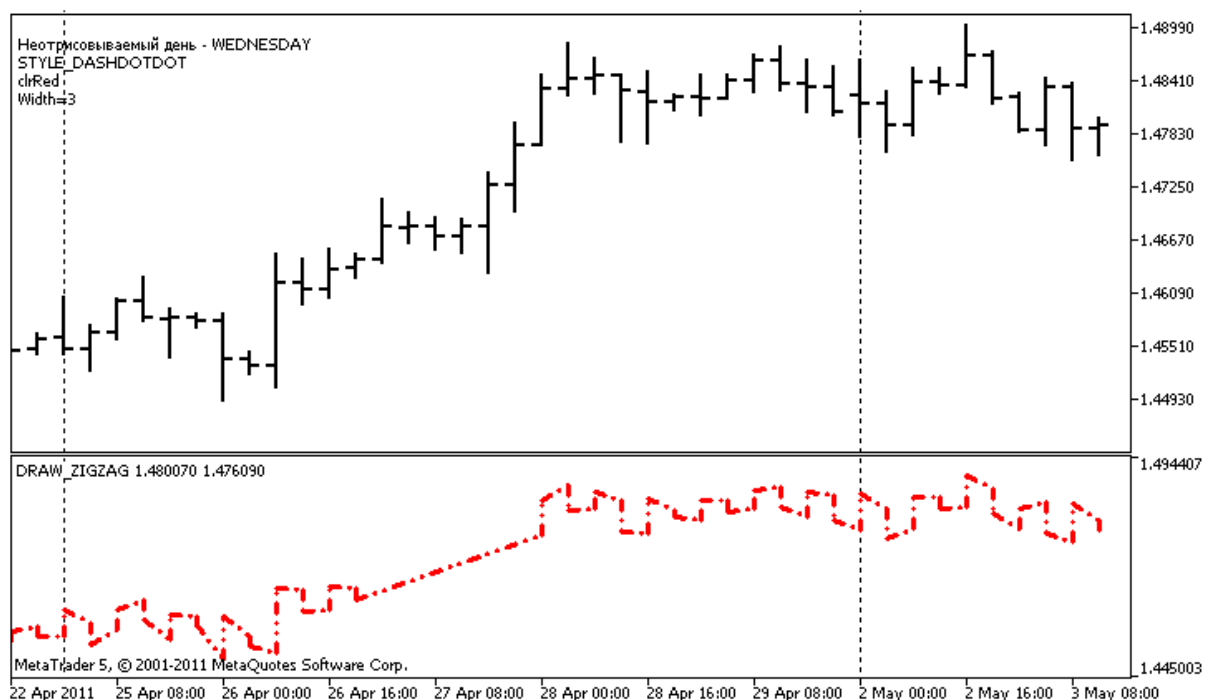
Die Segmente werden vom nicht leeren Wert eines Puffers zum nicht leeren Wert anderes Indikator-Puffers gezeichnet. Um anzugeben, welchen Wert als "leer" betrachtet werden soll, setzen Sie diesen Wert in der Eigenschaft [PLOT\\_EMPTY\\_VALUE](#):

```
//--- Der Leerer Wert (0) wird nicht in der Zeichnung teilnehmen
PlotIndexSetDouble(Index_der_Zeichnung_DRAW_ZIGZAG,PLOT_EMPTY_VALUE,0);
```

Immer füllen Sie die Indikator-Puffer mit Werten expliziert aus, geben Sie den leeren Wert in Puffer der Balken zu überspringen.

Die erforderliche Anzahl der Puffer für den Bau von DRAW\_ZIGZAG ist 2.

Ein Beispiel für einen Indikator, der eine Säge auf die High und Low Preise zeichnet. Farbe, Breite und Linienart des Zigzags werden nach dem Zufallsprinzip jede N Ticks geändert.



Bitte beachten Sie, dass für `plot1` mit dem DRAW\_ZIGZAG-Stil werden Eigenschaften mithilfe Compiler-Direktive [#property](#) angegeben, und dann in der Funktion [OnCalculate\(\)](#) werden diese drei Eigenschaften nach dem Zufallsprinzip angegeben. Der Parameter N wird in [externen Parametern](#) des

Indikators angegeben, damit kann es manuell angegeben werden (die Registerkarte "Einstellungen" im Indikatoreigenschaften-Fenster).

```
//+-----+
//|                                     DRAW_ZIGZAG.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Der Indikator demonstriert DRAW_ZIGZAG"
#property description "Zeichnet "Säge" aus den geraden Linien, überspringt Balken eines Tages"
#property description "Der Tag zu überspringen wird zufällig beim Start des Indikators"
#property description "Farbe, Breite und Stil der Segmente variieren zufällig"
#property description "jede N Ticks"

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots 1
//--- plot ZigZag
#property indicator_label1 "ZigZag"
#property indicator_type1  DRAW_ZIGZAG
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1 1

//--- Eingabeparameter
input int      N=5;           // Anzahl der Ticks für Änderung
//--- indicator buffers
double         ZigZagBuffer1[];
double         ZigZagBuffer2[];
//--- der Wochentag, für den der Indikator nicht gezeichnet wird
int invisible_day;
//--- Array, um die Farbe zu speichern
color colors[]={clrRed,clrBlue,clrGreen};
//--- Array für Speicherung der Linienstile
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOTDOT}

//+-----+
//| Custom indicator initialization function |
//+-----+

int OnInit()
{
//--- Bindung von Arrays und Indikator-Puffer
    SetIndexBuffer(0,ZigZagBuffer1,INDICATOR_DATA);
    SetIndexBuffer(1,ZigZagBuffer2,INDICATOR_DATA);
//--- Erhalten eine Zufallszahl von 0 bis 6, für diesen Tag wird der Indikator nicht gezeichnet
    invisible_day=MathRand()%6;
//--- Der Leerer Wert (0) wird nicht in der Zeichnung teilnehmen
```



```

    PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- Der Leerer Wert (0) wird nicht in der Zeichnung teilnehmen
    PlotIndexSetString(0,PLOT_LABEL,"ZigZag1;ZigZag2");
//---
    return(INIT_SUCCEEDED);
;}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
    static int ticks=0;
//--- Zählen wir Ticks, um den Stil, die Farbe und Breite der Linie zu ändern
    ticks++;
//--- Wenn eine ausreichende Anzahl von Ticks angesammelt hat
    if(ticks>=N)
    {
        //--- Ändern wir die Eigenschaften der Linie
        ChangeLineAppearance();
        //--- Setzen wir den Zähler der Ticks auf Null
        ticks=0;
    };
//--- Die Zeitstruktur ist benötigt, um den Wochentag jedes Balkens zu erhalten
    MqlDateTime dt;

//--- Startposition der Berechnungen
    int start=0;
//--- Wenn der Indikator wurde auf dem vorherigen Tick berechnet, dann beginnen wir Be
    if(prev_calculated!=0) start=prev_calculated-1;
//--- Berechnungszyklus
    for(int i=start;i<rates_total;i++)
    {
        //--- Schreiben die Balkenöffnungszeit in die Struktur
        TimeToStruct(time[i],dt);
        //--- Wenn der Wochentag dieses Balken invisible_day ist
        if(dt.day_of_week==invisible_day)
        {
            //--- Wir schreiben leere Werte des Puffers für diesen Balken
            ZigZagBuffer1[i]=0;

```

```

        ZigZagBuffer2[i]=0;
    };
    //--- Wenn der Wochentag entspricht, füllen wir die Puffer
else
{
    //--- Wenn die Nummer des Balkens ist gerade
    if(i%2==0)
    {
        //--- Wir schreiben High in den ersten Puffer und Low in den zweiten Puffer
        ZigZagBuffer1[i]=high[i];
        ZigZagBuffer2[i]=low[i];
    };
    //--- Die Nummer des Balkens ist ungerade
else
{
    //--- Wir füllen den Balken in der umgekehrten Reihenfolge
    ZigZagBuffer1[i]=low[i];
    ZigZagBuffer2[i]=high[i];
};
};
};

//--- return value of prev_calculated for next call
return(rates_total);
};

//+-----+
//| Ändert das Aussehen der Segmente in Zigzag |
//+-----+
void ChangeLineAppearance()
{
    //--- String um Informationen über Eigenschaften von ZigZag zu erstellen
    string comm="";
    //--- Block mit Änderungen der Farbe des ZigZags
    int number=MathRand(); // Erhalten wir Zufallszahl
    //--- Teiler der Zahl entspricht der Größe des Arrays colors[]
    int size=ArraySize(colors);
    //--- Erhalten wir den Index als den Rest der ganzzahligen Division, um eine neue Farbe
    int color_index=number%size;
    //--- Geben wir die Farbe als Eigenschaft PLOT_LINE_COLOR an
    PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
    //--- Schreiben wir die Linienfarbe
    comm=comm+"\r\n"+(string)colors[color_index];

    //--- Block mit Änderungen der Linienbreite
    number=MathRand();
    //--- Erhalten wir die Breite als den Rest der ganzzahligen Division
    int width=number%5; // Breite ist von 0 bis 4
    //--- Geben wir die Farbe als Eigenschaft PLOT_LINE_WIDTH
    PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
    //--- Schreiben wir die Linienbreite

```

```
comm=comm+"\r\nWidth="+IntegerToString(width);

//--- Block mit Änderungen des Linienstils
number=MathRand();
//--- Teiler der Zahl entspricht der Größe des Arrays styles
size=ArraySize(styles);
//--- Erhalten wir den Index als den Rest der ganzzahligen Division, um ein neuer Stil
int style_index=number%size;
//--- Geben wir die Farbe als Eigenschaft PLOT_LINE_COLOR an
PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- Schreiben wir den Linienstil
comm="\r\n"+EnumToString(styles[style_index])+" "+comm;
//--- Informationen über den Tag, der in den Berechnungen ignoriert wird, hinzufügen
comm="\r\nUngezeichneter Tag - "+EnumToString((ENUM_DAY_OF_WEEK)invisible_day)+comm;
//--- Zeigen wir Informationen auf dem Chart durch den Kommentar
Comment(comm);

; }
```

## DRAW\_FILLING

Stil DRAW\_FILLING zeichnet Farbbereich zwischen den Werten der zwei Indikator-Puffer. In der Tat zeichnet dieser Stil zwei Zeilen und füllt den Raum zwischen ihnen mit einem der zwei angegebenen Farben. Es wird für Indikatoren, die die Kanäle zeichnen, benutzt. Keiner der Puffer darf nicht nur leere Werte erhalten, da in diesem Fall auch für die Erbringung nicht der Fall ist.

Sie können zwei Farbe für Füllen angeben:

- die erste Farbe ist für die Bereiche, wo die Werte im ersten Indikatorpuffer größer als jene im zweiten Indikator-Puffer sind;
- die zweite Farbe ist für die Bereiche, wo die Werte im zweiten Indikatorpuffer größer als jene im ersten Indikator-Puffer sind.

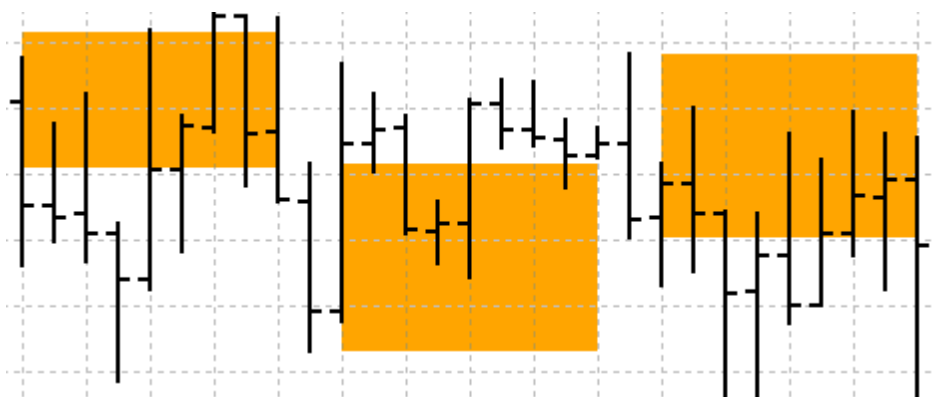
Füllfarbe kann durch [Compiler-Direktiven](#) oder dynamisch mit der [PlotIndexSetInteger\(\)](#)-Funktion angegeben werden. Dynamische Veränderungen in den Eigenschaften der graphischen Konstruktion ermöglichen es Ihnen, die Indikatoren zu "wiederbeleben" so, dass sie ihr Aussehen ändern je nach der aktuellen Situation.

Der Indikator wird für alle Bars, für die die Werte der beiden Indikatorpuffer nicht gleich 0 sind und nicht gleich dem leeren Wert sind, berechnet. Um anzugeben, welcher Wert als "leer" betrachtet werden sollte, setzen Sie diesen Wert in Eigenschaft [PLOT\\_EMPTY\\_VALUE](#):

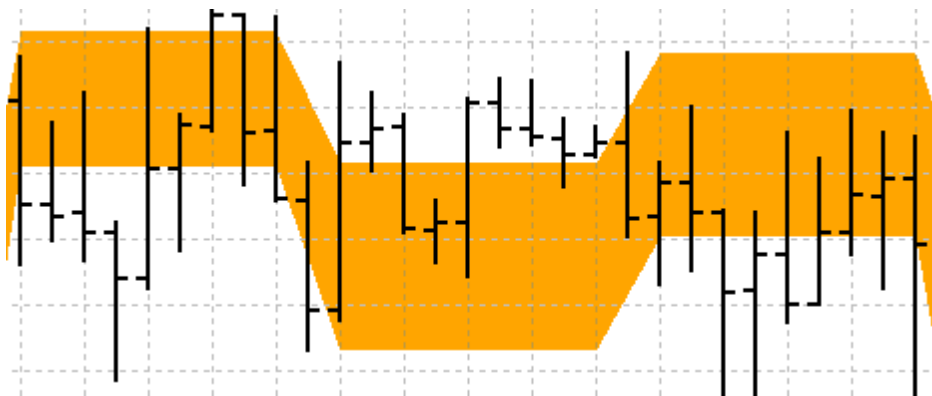
```
#define INDICATOR_EMPTY_VALUE -1.0
...
//--- Wert INDICATOR_EMPTY_VALUE (leerer Wert) wird nicht in Berechnungen teil nehmen
PlotIndexSetDouble(Plot_Index_DRAW_FILLING, PLOT_EMPTY_VALUE, INDICATOR_EMPTY_VALUE);
```

Zeichnung auf die Bars, die nicht in die Berechnung des Indikators beteiligt sind, sind von den Werten in den Indikatorpuffern abhängig:

- Die Bars, für die die Werte der beiden Indikatorpuffer 0 sind, beteiligen nicht in der Zeichnung des Indikators. Das ist ein Bereich mit Null-Werten wird nicht übermalt werden.



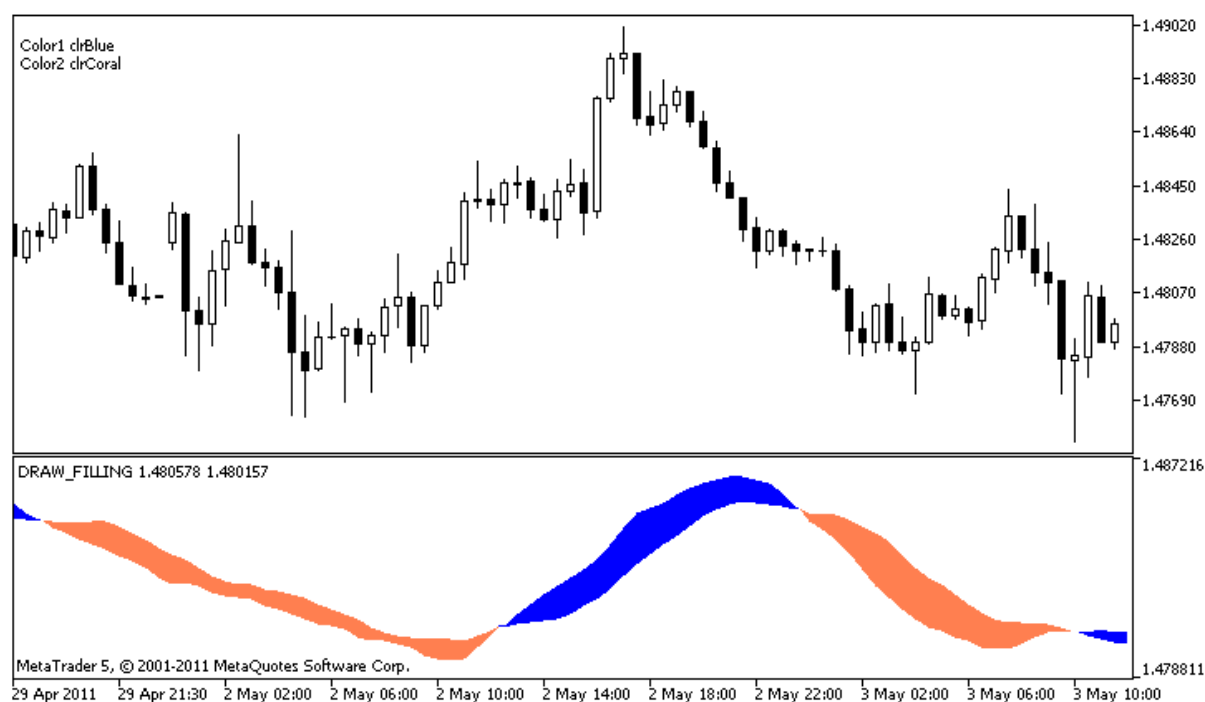
- Die Bars, für die die Werte der Indikatorpuffer dem leeren Wert gleich sind, beteiligen in der Zeichnung des Indikators. Bereich mit leeren Werten wird übermalt, um die Bereiche mit sinnvollen Werten zu verbinden.



Es ist wichtig zu beachten, dass, wenn ein "leerer Wert" Null gleich ist, dann werden die Bars, die nicht in die Berechnung des Indikators beteiligen, auch übermalt.

Die erforderliche Anzahl der Puffer für den Bau von DRAW\_FILLING ist 2.

Ein Beispiel für einen Indikator, der einen Kanal zwischen zwei MAs mit verschiedenen Perioden der Mitteilung in einem separaten Fenster zeichnet. Farbwechsel beim Überschreiten der MAs zeigt Veränderung der aufsteigenden und absteigenden Trends. Farben variieren zufällig jede N Ticks. Der Parameter N wird in [externen Parameter](#) des Indikators angegeben, damit kann es manuell angegeben werden (die Registerkarte "Einstellungen" im Indikatoreigenschaften-Fenster).



Bitte beachten Sie, dass für `plot1` mit dem `DRAW_FILLING`-Stil werden zwei Farben mithilfe Compiler-Direktive `#property` angegeben, und dann in der Funktion [OnCalculate\(\)](#) werden neue Farben nach dem Zufallsprinzip angegeben.

```
//+-----+
//|                                     DRAW_FILLING.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
```

```

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Der Indikator demonstriert DRAW_FILLING"
#property description "Zeichnet in einem separaten Fenster einen Kanal zwischen zwei MA"
#property description "Füllfarbe des Kanals variiert zufällig"
#property description "jede N Ticks verändert"

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots 1
//--- plot Intersection
#property indicator_label1 "Intersection"
#property indicator_type1  DRAW_FILLING
#property indicator_color1  clrRed,clrBlue
#property indicator_width1 1
//--- Eingabeparameter
input int      Fast=13;          // Periode der schnellen MA
input int      Slow=21;          // Periode der schnellen MA
input int      shift=1;          // Verschiebung nach Zukunft (positive)
input int      N=5;              // Anzahl der Ticks für Änderung
//--- Indicator-Puffer
double         IntersectionBuffer1[];
double         IntersectionBuffer2[];
int fast_handle;
int slow_handle;
//--- Array, um die Farbe zu speichern
color colors[]={clrRed,clrBlue,clrGreen,clrAquamarine,clrBlanchedAlmond,clrBrown,clrC
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- indicator buffers mapping
    SetIndexBuffer(0,IntersectionBuffer1,INDICATOR_DATA);
    SetIndexBuffer(1,IntersectionBuffer2,INDICATOR_DATA);
//---
    PlotIndexSetInteger(0,PLOT_SHIFT,shift);
//---
    fast_handle=iMA(_Symbol,_Period,Fast,0,MODE_SMA,PRICE_CLOSE);
    slow_handle=iMA(_Symbol,_Period,Slow,0,MODE_SMA,PRICE_CLOSE);
//---
    return(INIT_SUCCEEDED);
};
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,

```

```

        const int prev_calculated,
        const datetime &time[],
        const double &open[],
        const double &high[],
        const double &low[],
        const double &close[],
        const long &tick_volume[],
        const long &volume[],
        const int &spread[])

{
    static int ticks=0;
    //--- Zählen wir Ticks, um den Stil, die Farbe und Breite der Linie zu ändern
    ticks++;
    //--- Wenn eine ausreichende Anzahl von Ticks angesammelt hat
    if(ticks>=N)
    {
        //--- Ändern wir die Eigenschaften der Linie
        ChangeLineAppearance();
        //--- Setzen wir den Zähler der Ticks auf Null
        ticks=0;
    }

    //--- Die erste Berechnung des Indikators, oder Daten haben geändert haben und eine v
    if(prev_calculated==0)
    {
        //--- Kopieren wir alle Werte der Indikatoren in die entsprechende Puffer
        int copied1=CopyBuffer(fast_handle,0,0,rates_total,IntersectionBuffer1);
        int copied2=CopyBuffer(slow_handle,0,0,rates_total,IntersectionBuffer2);
    }
    else // sparsam füllen nur die Daten, die aktualisiert wurden
    {
        //--- Erhalten wir die Differenz in Balken zwischen aktuellen und früheren Lauf
        int to_copy=rates_total-prev_calculated;
        //--- Wenn es keinen Unterschied gibt, kopieren wir nur einen Wert - das Wert an
        if(to_copy==0) to_copy=1;
        //--- Kopieren to_copy Werte an das Ende der Indikator-Puffer
        int copied1=CopyBuffer(fast_handle,0,0,to_copy,IntersectionBuffer1);
        int copied2=CopyBuffer(slow_handle,0,0,to_copy,IntersectionBuffer2);
    }

    //--- return value of prev_calculated for next call
    return(rates_total);
}

//+-----+
//| Ändert die Füllfarbe des Kanals |
//+-----+
void ChangeLineAppearance()
{
    //--- String um Informationen über die Eigenschaften der Linie zu erstellen
    string comm="";

```

```
//--- Block mit Änderungen der Linienfarbe
    int number=MathRand(); // Erhalten wir Zufallszahl
//--- Teiler der Zahl entspricht der Größe des Arrays colors[]
    int size=ArraySize(colors);

//--- Erhalten wir den Index als den Rest der ganzzahligen Division, um eine neue Farbe
    int color_index1=number%size;
//--- Geben wir die erste Farbe als Eigenschaft PLOT_LINE_COLOR
    PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,colors[color_index1]);
//--- Schreiben wir die erste Farbe
    comm=comm+"\r\nColor1 "+(string)colors[color_index1];

//--- Erhalten wir den Index als den Rest der ganzzahligen Division, um eine neue Farbe
    number=MathRand(); // Erhalten eine Zufallszahl
    int color_index2=number%size;
//--- Geben wir die zweite Farbe als Eigenschaft PLOT_LINE_COLOR
    PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,colors[color_index2]);
//--- Schreiben wir die zweite Farbe
    comm=comm+"\r\nColor2 "+(string)colors[color_index2];
//--- Zeigen wir Informationen auf dem Chart durch den Kommentar
    Comment(comm);

; }
```



## DRAW\_BARS

DRAW\_BARS-Stil zeichnet Balken auf den Werten der vier Indikator-Puffer, die die Preise Open, High, Low und Close enthalten. Es erlaubt individuelle Indikatoren in Form von Balken, darunter ein in separates Grafikfenster und andere Finanzinstrumente, zu erstellen.

Balkenfarbe kann durch [Compiler-Direktiven](#) oder dynamisch mit der [PlotIndexSetInteger\(\)](#)-Funktion angegeben werden. Dynamische Veränderungen in den Eigenschaften der graphischen Konstruktion ermöglichen es Ihnen, die Indikatoren zu "wiederbeleben" so, dass sie ihr Aussehen ändern je nach der aktuellen Situation.

Der Indikator ist nur für die Balken, für die nicht leer Werte **aller** vier Indikator-Puffern angegeben wird, gezeichnet. Um anzugeben, welchen Wert als "leer" betrachtet werden soll, setzen Sie diesen Wert in der Eigenschaft [PLOT\\_EMPTY\\_VALUE](#):

```
//--- Der Leerer Wert (0) wird nicht in der Zeichnung teilnehmen
PlotIndexSetDoubleIndex_der_Darstellung_DRAW_BARS, PLOT_EMPTY_VALUE, 0);
```

Immer füllen Sie die Indikator-Puffer mit Werten expliziert aus, geben Sie den leeren Wert in Puffer der Balken zu überspringen.

Die erforderliche Anzahl der Puffer für den Bau von DRAW\_BARS ist 4. Alle Puffer sollen eine nach der anderen in dieser Reihenfolge gehen: Open, High, Low und Close. Keiner der Puffer darf nicht nur leere Werte erhalten, da in diesem Fall nichts gezeichnet wird.

Ein Beispiel für einen Indikator, der Balken des Finanzinstrumenten in einem separaten Fenster zeichnet. Die Balkenfarbe wird zufällig alle N Ticks verändert. Der Parameter N wird in [externen Parameter](#) des Indikators angegeben, damit kann es manuell angegeben werden (die Registerkarte "Einstellungen" im Indikatoreigenschaften-Fenster).



Bitte beachten Sie, dass für plot1 mit dem DRAW\_BARS-Stil die Farbe mit Compiler-Direktiven [#property](#) angegeben wird, und dann in der Funktion [OnCalculate\(\)](#) wird die Farbe nach dem

Zufallsprinzip aus einer vorbereiteten Liste ausgewählt.

```
//+-----+
//|                                     DRAW_BARS.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Der Indikator zeigt DRAW_BARS"
#property description "Zeichnet in einem separaten Fenster Balken des ausgewählten Syn
#property description "Die Farbe und Größe der Balken und der Symbol werden nach dem ?
#property description "jede N Ticks verändert

#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   1
//--- plot Bars
#property indicator_label1  "Bars"
#property indicator_type1   DRAW_BARS
#property indicator_color1  clrGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- Eingabeparameter
input int      N=5;           // Anzahl der Ticks um Art zu ändern
input int      bars=500;      // Anzahl der Balken zu zeigen
input bool     messages=false; // Ausgabe der Meldungen ins"Experts"-Journal
//--- Indicator-Puffer
double         BarsBuffer1[];
double         BarsBuffer2[];
double         BarsBuffer3[];
double         BarsBuffer4[];
//--- Symbolname
string symbol;
//--- Array, um die Farbe zu speichern
color colors[]={clrRed,clrBlue,clrGreen,clrPurple,clrBrown,clrIndianRed};
//+-----+
//| Custom indicator initialization function |
//+-----+

int OnInit()
{
//--- wenn es gibt zu wenig bars, Arbeit früher vollenden
if(bars<50)
{
    Comment("Geben Sie eine größere Anzahl von Bars! Arbeit des Indikators ist beend
    return(INIT_PARAMETERS_INCORRECT);
};
}
```

```

//--- indicator buffers mapping
SetIndexBuffer(0,BarsBuffer1,INDICATOR_DATA);
SetIndexBuffer(1,BarsBuffer2,INDICATOR_DATA);
SetIndexBuffer(2,BarsBuffer3,INDICATOR_DATA);
SetIndexBuffer(3,BarsBuffer4,INDICATOR_DATA);
//--- Name des Symbols, für das die Balken gezeichnet werden
symbol=_Symbol;
//--- Einstellen wir Anzeige des Symbols
PlotIndexSetString(0,PLOT_LABEL,symbol+" Open;"+symbol+" High;"+symbol+" Low;"+symbol+" Close;");
IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_BARS("+symbol+")");
//--- Leerer Wert
PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0.0);
//---
return(INIT_SUCCEEDED);
; }
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
    static int ticks=0;
//--- Zählen wir Ticks, um den Stil, die Farbe und Breite der Linie zu ändern
    ticks++;
//--- Wenn eine ausreichende Anzahl von Ticks angesammelt hat
    if(ticks>=N)
    {
//--- Wählen wir ein neues Symbol aus dem "Market Watch"
        symbol=GetRandomSymbolName();
//--- Ändern wir die Eigenschaften der Linie
        ChangeLineAppearance();

        int tries=0;
//--- Machen wir 5 Versuche, um den Puffer mit Preise aus symbol zu füllen
        while(!CopyFromSymbolToBuffers(symbol,rates_total) && tries<5)
        {
//--- Zähler der CopyFromSymbolToBuffers() Funktionsaufrufe
            tries++;
        };
//--- Setzen wir den Zähler der Ticks auf Null
        ticks=0;
    }
}

```

```

    ;}
//--- return value of prev_calculated for next call
    return(rates_total);
    ;}
//+-----+
//| Füllen wir den Indikator-Puffer mit Preise aus |
//+-----+
bool CopyFromSymbolToBuffers(string name,int total)
{
//--- In die Array rates[] kopieren wir Preise Open, High, Low und Close
    MqlRates rates[];
//--- Zähler der Versuche
    int attempts=0;
//--- Wieviel kopiert
    int copied=0;
//--- Machen wir 25 Versuche, um die Zeitreihe für das gewünschten Symbol zu erhalten
    while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
    {
        Sleep(100);
        attempts++;
        if(messages) PrintFormat("%s CopyRates(%s) attempts=%d",__FUNCTION__,name,attempts);
    }
//--- Wenn es gelang nicht eine ausreichende Anzahl von Balken zu kopieren
    if(copied!=bars)
    {
        //--- erstellen wir einen Nachrichtenstring
        string comm=StringFormat("Für Symbol %s konnte nur %d Balken von %d gefragten Ba
                                name,
                                copied,
                                bars
                                );

        //--- Zeigen wir einer Nachricht in einer Kommentar auf dem Haupt-Chart-Fenster
        Comment(comm);
        //--- Anzeige von Informationen
        if(messages) Print(comm);
        return(false);
    }
else
{
    //--- Einstellen wir Anzeige des Symbols
    PlotIndexSetString(0,PLOT_LABEL,name+" Open;" +name+" High;" +name+" Low;" +name+"
    IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_BARS (" +name+" )");
}
//--- Initialisieren wir den Puffer leere Werte
    ArrayInitialize(BarsBuffer1,0.0);
    ArrayInitialize(BarsBuffer2,0.0);
    ArrayInitialize(BarsBuffer3,0.0);
    ArrayInitialize(BarsBuffer4,0.0);
//--- Kopieren wir Preise in Puffer

```

```

    for(int i=0;i<copied;i++)
    {
//--- Berechnen wir den entsprechenden Index für den Puffer
        int buffer_index=total-copied+i;
//--- Schreiben wir die Preise in Puffer
        BarsBuffer1[buffer_index]=rates[i].open;
        BarsBuffer2[buffer_index]=rates[i].high;
        BarsBuffer3[buffer_index]=rates[i].low;
        BarsBuffer4[buffer_index]=rates[i].close;
    };
    return(true);
};

//+-----+
//| Gibt ein zufälliges Symbol aus dem Market Watch zurück |
//+-----+
string GetRandomSymbolName()
{
//--- Anzahl der in dem "Market Watch" Fenster angezeigten Symbole
    int symbols=SymbolsTotal(true);
//--- Position des Symbols in der Liste - eine Zufallszahl zwischen 0 und symbols
    int number=MathRand()%symbols;
//--- Gibt den Namen des Symbols an der angegebenen Position zurück
    return SymbolName(number,true);
};

//+-----+
//| Ändert das Aussehen von Balken |
//+-----+
void ChangeLineAppearance()
{
//--- String um Informationen über die Eigenschaften der Balken zu erstellen
    string comm="";
//--- Block mit Änderungen der Balkenfarbe
    int number=MathRand(); // Erhalten wir Zufallszahl
//--- Teiler der Zahl entspricht der Größe des Arrays colors[]
    int size=ArraySize(colors);
//--- Erhalten wir den Index als den Rest der ganzzahligen Division, um eine neue Farbe
    int color_index=number%size;
//--- Geben wir die Farbe als Eigenschaft PLOT_LINE_COLOR an
    PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- Schreiben wir die Linienfarbe
    comm=comm+"\r\n"+(string)colors[color_index];

//--- Block mit Änderungen der Balkenbreite
    number=MathRand();
//--- Erhalten wir die Breite als den Rest der ganzzahligen Division
    int width=number%5; // Breite ist von 0 bis 4
//--- Geben wir die Farbe als Eigenschaft PLOT_LINE_WIDTH
    PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Schreiben wir die Linienbreite

```

```
comm=comm+"\r\nWidth="+IntegerToString(width);

//--- Schreiben wir den Namen des Symbols
comm="\r\n"+symbol+comm;

//--- Zeigen wir Informationen auf dem Chart durch den Kommentar
Comment(comm);

; }
```

## DRAW\_CANDLES

DRAW\_CANDLES-Stil zeichnet Kerzen auf den Werten der vier Indikator-Puffer, die die Preise Open, High, Low und Close enthalten. Es erlaubt individuelle Indikatoren in Form von Kerzen, darunter ein in separates Grafikfenster und andere Finanzinstrumente, zu erstellen.

Kerzenfarbe kann durch [Compiler-Direktiven](#) oder dynamisch mit der [PlotIndexSetInteger\(\)](#)-Funktion angegeben werden. Dynamische Veränderungen in den Eigenschaften der graphischen Konstruktion ermöglichen es Ihnen, die Indikatoren zu "wiederbeleben" so, dass sie ihr Aussehen ändern je nach der aktuellen Situation.

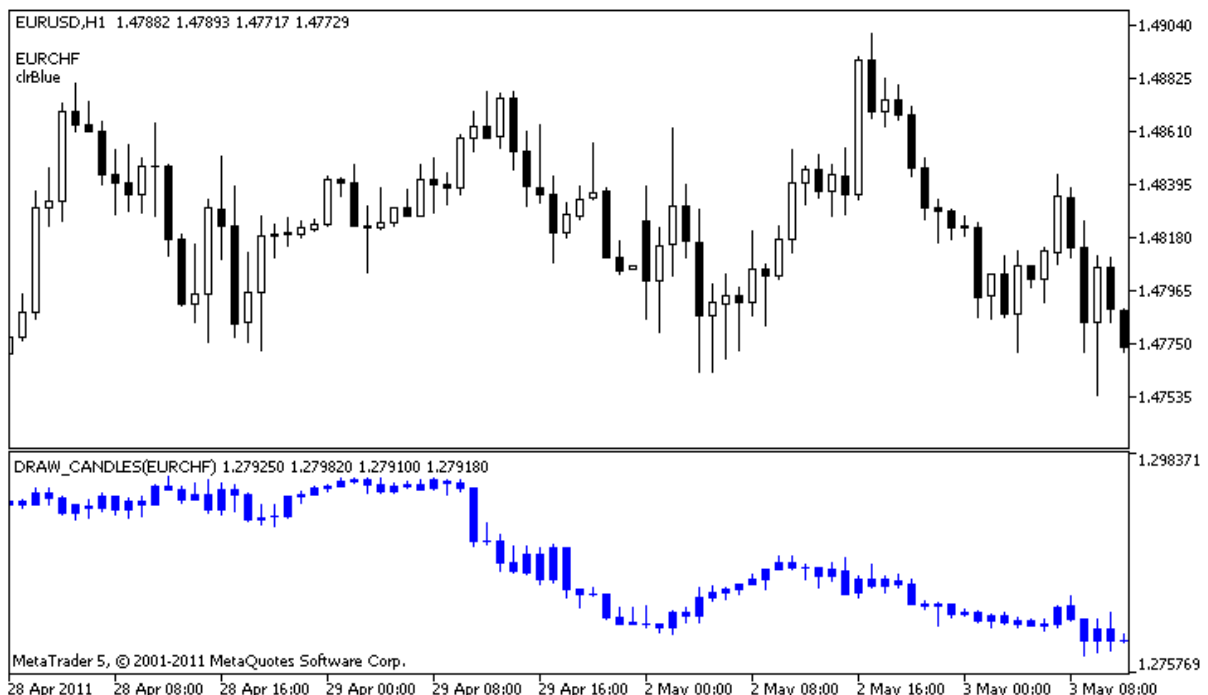
Der Indikator ist nur für die Balken, für die nicht leer Werte **aller** vier Indikator-Puffern angegeben wird, gezeichnet. Um anzugeben, welchen Wert als "leer" betrachtet werden soll, setzen Sie diesen Wert in der Eigenschaft [PLOT\\_EMPTY\\_VALUE](#):

```
//--- Der Leerer Wert (0) wird nicht in der Zeichnung teilnehmen
PlotIndexSetDouble(Index_der_Darstellung_DRAW_CANDLES, PLOT_EMPTY_VALUE, 0);
```

Immer füllen Sie die Indikator-Puffer mit Werten expliziert aus, geben Sie den leeren Wert in Puffer der Balken zu überspringen.

Die erforderliche Anzahl der Puffer für den Bau von DRAW\_CANDLES ist 4. Alle Puffer sollen eine nach der anderen in dieser Reihenfolge gehen: Open, High, Low und Close. Keiner der Puffer darf nicht nur leere Werte erhalten, da in diesem Fall nichts gezeichnet wird.

Ein Beispiel für einen Indikator, der Kerzen des Finanzinstrumenten in einem separaten Fenster zeichnet. Die Kerzenfarbe wird zufällig alle N Ticks verändert. Der Parameter N wird in [externen Parameter](#) des Indikators angegeben, damit kann es manuell angegeben werden (die Registerkarte "Einstellungen" im Indikatoreigenschaften-Fenster).



Bitte beachten Sie, dass für Plot1 die Farbe mit Compiler-Direktiven [#property](#) angegeben wird, und dann in der Funktion [OnCalculate\(\)](#) wird die Farbe nach dem Zufallsprinzip aus einer vorbereiteten

Liste ausgewählt.

```
//+-----+
//|                                     DRAW_CANDLES.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Der Indikator zeigt DRAW_CANDLES."
#property description "Zeichnet in einem separaten Fenster verschiedenfarbige Kerzen c
#property description " "
#property description "Die Farbe und Größe der Kerzen und der Symbol werden nach dem 2
#property description "jede N Ticks verändert."

#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots 1
//--- plot Bars
#property indicator_label1 "DRAW_CANDLES1"
#property indicator_type1  DRAW_CANDLES
#property indicator_color1  clrGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1 1

//--- Eingabeparameter
input int      N=5;           // Anzahl der Ticks um Art zu ändern
input int      bars=500;      // Anzahl der Balken zu zeigen
input bool     messages=false; // Ausgabe der Meldungen ins"Experts"-Journal
//--- Indicator-Puffer
double         Candle1Buffer1[];
double         Candle1Buffer2[];
double         Candle1Buffer3[];
double         Candle1Buffer4[];
//--- Symbolname
string symbol;
//--- Array, um die Farbe zu speichern
color colors[]={clrRed,clrBlue,clrGreen,clrPurple,clrBrown,clrIndianRed};
//+-----+
//| Custom indicator initialization function |
//+-----+

int OnInit()
{
//--- wenn es gibt zu wenig bars, Arbeit früher vollenden
if(bars<50)
{
Comment("Geben Sie eine größere Anzahl von Bars! Arbeit des Indikators ist beend
```



```

        return(INIT_PARAMETERS_INCORRECT);
    };
}

//--- indicator buffers mapping
SetIndexBuffer(0,Candle1Buffer1,INDICATOR_DATA);
SetIndexBuffer(1,Candle1Buffer2,INDICATOR_DATA);
SetIndexBuffer(2,Candle1Buffer3,INDICATOR_DATA);
SetIndexBuffer(3,Candle1Buffer4,INDICATOR_DATA);
//--- Leerer Wert
PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- Name des Symbols, für das die Balken gezeichnet werden
symbol=_Symbol;
//--- Einstellen wir Anzeige des Symbols
PlotIndexSetString(0,PLOT_LABEL,symbol+" Open;" +symbol+" High;" +symbol+" Low;" +symbol+" Close;");
IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_CANDLES (" +symbol + ")");
//---
return(INIT_SUCCEEDED);
};
}

//+-----+
//| Custom indicator iteration function |
//+-----+

int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
    static int ticks=INT_MAX-100;
    //--- Zählen wir Ticks, um den Stil, die Farbe und Breite der Linie zu ändern
    ticks++;
    //--- Wenn eine ausreichende Anzahl von Ticks angesammelt hat
    if(ticks>=N)
    {
        //--- Wählen wir ein neues Symbol aus dem "Market Watch"
        symbol=GetRandomSymbolName();
        //--- Ansicht verändern
        ChangeLineAppearance();
        //--- Wählen wir ein neues Symbol aus dem "Market Watch"
        int tries=0;
        //--- Machen wir 5 Versuche, um den Puffer plot1 mit Preise aus symbol zu füllen
        while(!CopyFromSymbolToBuffers(symbol,rates_total,0,
                                       Candle1Buffer1,Candle1Buffer2,Candle1Buffer3,Candle1Buffer4)
            && tries<5)
        {
            //--- Zähler der CopyFromSymbolToBuffers() Funktionsaufrufe

```

```

        tries++;
    };
    //--- Setzen wir den Zähler der Ticks auf Null
    ticks=0;
};
//--- return value of prev_calculated for next call
    return(rates_total);
};
//+-----+
//| Füllt die angegebene Kerze |
//+-----+
bool CopyFromSymbolToBuffers(string name,
                             int total,
                             int plot_index,
                             double &buff1[],
                             double &buff2[],
                             double &buff3[],
                             double &buff4[]
                             )
{
    //--- In die Array rates[] kopieren wir Preise Open, High, Low und Close
    MqlRates rates[];
    //--- Zähler der Versuche
    int attempts=0;
    //--- Wieviel kopiert
    int copied=0;
    //--- Machen wir 25 Versuche, um die Zeitreihe für das gewünschten Symbol zu erhalten
    while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
    {
        Sleep(100);
        attempts++;
        if(messages) PrintFormat("%s CopyRates(%s) attempts=%d",__FUNCTION__,name,attempts);
    };
    //--- Wenn es gelang nicht eine ausreichende Anzahl von Balken zu kopieren
    if(copied!=bars)
    {
        //--- erstellen wir einen Nachrichtenstring
        string comm=StringFormat("Für Symbol %s konnte nur %d Balken von %d gefragten Balken kopiert werden",
                                name,
                                copied,
                                bars
                                );
        //--- Zeigen wir einer Nachricht in einer Kommentar auf dem Haupt-Chart-Fenster
        Comment(comm);
        //--- Anzeige von Informationen
        if(messages) Print(comm);
        return(false);
    };
    else

```

```

    {
        //--- Einstellen wir Anzeige des Symbols
        PlotIndexSetString(plot_index,PLOT_LABEL,name+" Open;" +name+" High;" +name+" Low;
    };
//--- Initialisieren wir den Puffer leere Werte
ArrayInitialize(buff1,0.0);
ArrayInitialize(buff2,0.0);
ArrayInitialize(buff3,0.0);
ArrayInitialize(buff4,0.0);
//--- Kopieren wir Preise in den Puffer bei jedem Tick
for(int i=0;i<copied;i++)
    {
        //--- Berechnen wir den entsprechenden Index für den Puffer
        int buffer_index=total-copied+i;
        //--- Schreiben wir die Preise in Puffer
        buff1[buffer_index]=rates[i].open;
        buff2[buffer_index]=rates[i].high;
        buff3[buffer_index]=rates[i].low;
        buff4[buffer_index]=rates[i].close;
    };
    return(true);
};
//+-----+
//| Gibt ein zufälliges Symbol aus dem Market Watch zurück |
//+-----+
string GetRandomSymbolName()
{
    //--- Anzahl der in dem "Market Watch" Fenster angezeigten Symbole
    int symbols=SymbolsTotal(true);
    //--- Position des Symbols in der Liste - eine Zufallszahl zwischen 0 und symbols
    int number=MathRand()%symbols;
    //--- Gibt den Namen des Symbols an der angegebenen Position zurück
    return SymbolName(number,true);
};
//+-----+
//| Ändert das Aussehen von Balken |
//+-----+
void ChangeLineAppearance()
{
    //--- String um Informationen über die Eigenschaften der Balken zu erstellen
    string comm="";
    //--- Block mit Änderungen der Balkenfarbe
    int number=MathRand(); // Erhalten wir Zufallszahl
    //--- Teiler der Zahl entspricht der Größe des Arrays colors[]
    int size=ArraySize(colors);
    //--- Erhalten wir den Index als den Rest der ganzzahligen Division, um eine neue Farbe
    int color_index=number%size;
    //--- Geben wir die Farbe als Eigenschaft PLOT_LINE_COLOR an
    PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);

```

```
//--- Schreiben wir die Farbe
    comm=comm+"\r\n"+(string)colors[color_index];
//--- Schreiben wir den Namen des Symbols
    comm+"\r\n"+symbol+comm;
//--- Zeigen wir Informationen auf dem Chart durch den Kommentar
    Comment(comm);
; }
```

## DRAW\_COLOR\_LINE

Stil DRAW\_COLOR\_LINE ist eine Farbversion von [DRAW\\_LINE](#), er zeichnet eine Linie von den Werten des Indikator-Puffers. Aber dieser Stil, wie in allen farbigen Stile, mit einem Namen **COLOR**, hat auch ein besonderer Indikator-Puffer, der den Index (Nummer) der Farbe aus dem angegebenen Farbarray erhielt. So kann die Farbe jedes Liniensegment eingestellt werden, wenn Sie die Farbindex der Linie auf diesem Balken angeben.

Die Breite, Stil und Farben um die Linie zu färben kann durch [Compiler-Direktiven](#) oder dynamisch mit der [PlotIndexSetInteger\(\)](#)-Funktion angegeben werden. Dynamische Veränderungen in den Eigenschaften der graphischen Konstruktion ermöglichen es Ihnen, die Indikatoren zu "wiederbeleben" so, dass sie ihr Aussehen ändern je nach der aktuellen Situation.

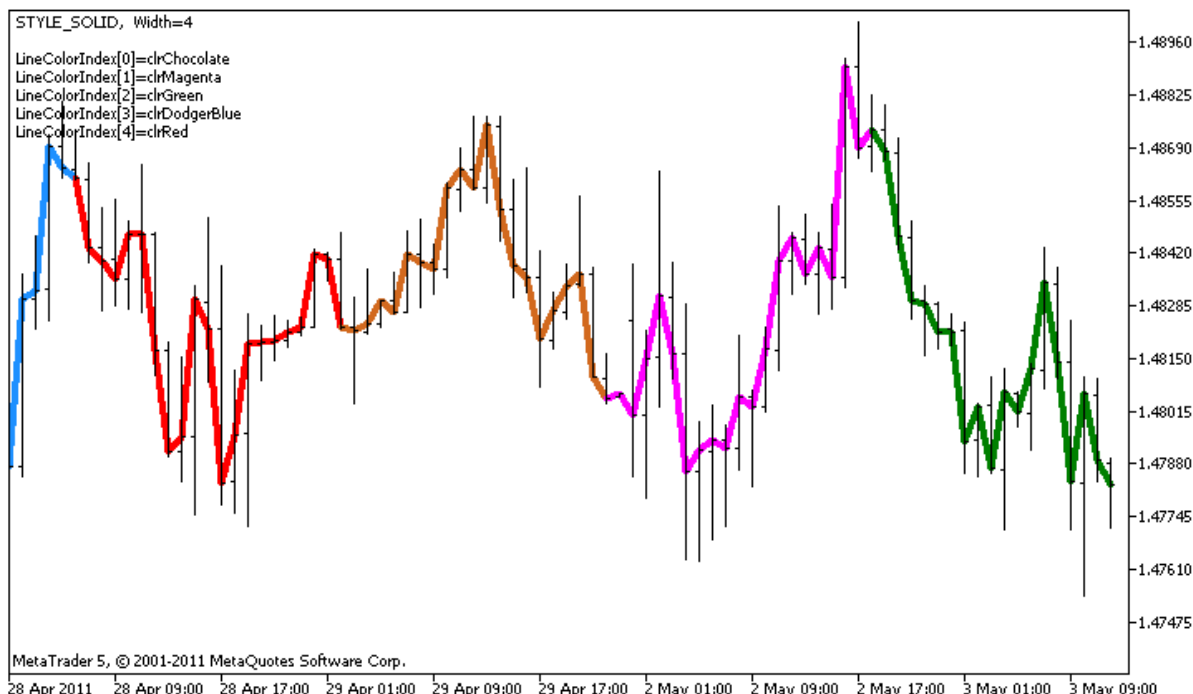
Die erforderliche Anzahl der Puffer für den Bau von DRAW\_COLOR\_LINE ist 2.

- einen Puffer für Speicherung die Werte des Indikators, auf denen die Linie gezeichnet wird;
- einen Puffer, um die Farbindex der Linie auf jedem Balken.

Farben können mit der Compiler-Direktive `#property indicator_color1` durch Kommas angegeben werden. Die Anzahl der Farben darf maximal 64 sein.

```
//--- Definieren wir 5 Farben um Balken zu färben (sie sind im speziellen Array gespeichert)
#property indicator_color1 clrRed,clrBlue,clrGreen,clrOrange,clrDeepPink // (Sie können weitere Farben angeben)
```

Ein Beispiel für einen Indikator, der eine Linie auf die Close Preise der Balken zeichnet. Breite und Stil der Linie werden nach dem Zufallsprinzip jede N=5 Ticks geändert.



Farben der Liniensegmente werden nach dem Zufallsprinzip in der benutzerdefinierten Funktion `ChangeColors()` geändert.

```
//+-----+
//| Ändert die Farbe der Linienteile |
```

```
//+-----+
void ChangeColors(color &cols[],int plot_colors)
{
//--- Anzahl der Farben
    int size=ArraySize(cols);
//---
    string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- Für jede Farbeindex definieren wir eine neue Farbe zufällig
    for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
    {
        //--- erhalten wir eine Zufallszahl
        int number=MathRand();
        //--- Erhalten wir den Index in Array col[] als den Rest der ganzzahligen Divisi
        int i=number%size;
        //--- Geben wir die Farbe für jeden Index als Eigenschaft PLOT_LINE_COLOR an
        PlotIndexSetInteger(0, // Nummer der Grafikstile
                             PLOT_LINE_COLOR, // Property Identifier
                             plot_color_ind, // Farbeindex, in dem wir die Farbe s
                             cols[i]); // neue Farbe

        //--- schreiben wir die Farbe
        comm=comm+StringFormat("LineColorIndex[%d]=%s \r\n",plot_color_ind,ColorToString
        ChartSetString(0,CHART_COMMENT,comm);
    }
//---
};
```

Das Beispiel zeigt die Besonderheit der Farbe-Versionen der Indikatoren - um die Farbe des Liniensegments zu ändern, brauchen Sie nicht die Werte im Puffer ColorLineColors[] (die Farbeindizes enthält) zu ändern. Es genügt, neue Farben in einem speziellen Array zu definieren. Dies ermöglicht Ihnen, schnell die Farbe für die grafische Konstruktion zu ändern, indem Sie nur eine kleine Array von Farben mit der [PlotIndexSetInteger\(\)](#) Funktion ändern.

Bitte beachten Sie, dass für plot1 mit dem DRAW\_COLOR\_LINE-Stil werden Eigenschaften mithilfe Compiler-Direktive [#property](#) angegeben, und dann in der Funktion [OnCalculate\(\)](#) werden diese drei Eigenschaften nach dem Zufallsprinzip angegeben.

Die Parameter N und Länge (Länge der farbigen Segmente der Balken) sind in [externen Parametern](#) des Indikators angegeben, damit kann es manuell angegeben werden (die Registerkarte "Einstellungen" im Indikatoreigenschaften-Fenster).

```
//+-----+
//|                                     DRAW_COLOR_LINE.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

```

#property description "Der Indikator demonstriert DRAW_COLOR_LINE"
#property description "Zeichnet eine Linie mit farbigen Segmenten von 20 Balken auf C
#property description "Breite, Stil und Farbe der Liniensegmente werden zufällig"
#property description "jede N Ticks verändert

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots 1
//--- plot ColorLine
#property indicator_label1 "ColorLine"
#property indicator_type1 DRAW_COLOR_LINE
//--- Definieren wir 5 Farben um Balken zu färben (sie sind im speziellen Array gespei
#property indicator_color1 clrRed,clrBlue,clrGreen,clrOrange,clrDeepPink // (Sie könn
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//--- Eingabeparameter
input int N=5; // Anzahl der Ticks für Änderung
input int Length=20; // Länge der Farbsegmente in Balken
int line_colors=5; // Anzahl der angegebenen Farben ist 5 - Sehen Sie #prop
//--- Buffer für das Rendering
double ColorLineBuffer[];
//--- Puffer zu speichern Farben der Liniensegmente auf jedem Balken
double ColorLineColors[];

//--- Array, um die Farbe zu speichern, enthält 7 Elemente
color colors[]={clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGolde
//--- Array für Speicherung der Linienstile
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- Bindung vom Array und Indikator-Puffer
SetIndexBuffer(0,ColorLineBuffer,INDICATOR_DATA);
SetIndexBuffer(1,ColorLineColors,INDICATOR_COLOR_INDEX);
//--- Initialisierung des Zufallszahlengenerators
MathSrand(GetTickCount());
//---
return(INIT_SUCCEEDED);
;}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],

```

```

        const double &low[],
        const double &close[],
        const long &tick_volume[],
        const long &volume[],
        const int &spread[])

{
    static int ticks=0;
//--- Zählen wir Ticks, um den Stil, die Farbe und Breite der Linie zu ändern
    ticks++;
//--- Wenn eine kritische Anzahl von Ticks angesammelt hat
    if(ticks>=N)
    {
        //--- Ändern wir die Eigenschaften der Linie
        ChangeLineAppearance();
//--- Ändern wir die Farben der Liniensegmente
        ChangeColors(colors,5);
        //--- Setzen wir den Zähler der Ticks auf Null
        ticks=0;
    };

//--- Block mit Berechnung der Indikatorwerte
    for(int i=0;i<rates_total;i++)
    {
        //--- Schreiben wir den Wert des Indikators in den Puffer
        ColorLineBuffer[i]=close[i];
        //--- Nun definieren wir eine zufällige Farbeindex für diesen Balken
        int color_index=i%(5*Length);
        color_index=color_index/Length;
        //--- Auf diesem balken wird die Linie mit der Farbe Nummer color_index gezeichnet
        ColorLineColors[i]=color_index;
    };

//--- den Wert prev_calculated für den nächsten Anruf der Funktion zurückgeben
    return(rates_total);
};

//+-----+
//| Ändert die Farbe der Linienteile |
//+-----+
void ChangeColors(color &cols[],int plot_colors)
{
    //--- Anzahl der Farben
    int size=ArraySize(cols);
//---
    string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- Für jede Farbeindex definieren wir eine neue Farbe zufällig
    for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
    {
        //--- erhalten wir eine Zufallszahl

```



```

    int number=MathRand();
    //--- Erhalten wir den Index in Array col[] als den Rest der ganzzahligen Division
    int i=number%size;
    //--- Geben wir die Farbe für jeden Index als Eigenschaft PLOT_LINE_COLOR an
    PlotIndexSetInteger(0, // Nummer der Grafikstile
        PLOT_LINE_COLOR, // Property Identifier
        plot_color_ind, // Farbeindex, in dem wir die Farbe setzen
        cols[i]); // neue Farbe

    //--- schreiben wir die Farbe
    comm=comm+StringFormat("LineColorIndex[%d]=%s \r\n",plot_color_ind,ColorToString(cols[i]));
    ChartSetString(0,CHART_COMMENT,comm);
; }

//---
; }

//+-----+
//| Ändert das Aussehen der angezeigten Linie im Indikator |
//+-----+

void ChangeLineAppearance()
{
    //--- String um Informationen über die Eigenschaften der Linie zu erstellen
    string comm="";
    //--- Block mit Änderungen der Linienbreite
    int number=MathRand();
    //--- Erhalten wir die Breite als den Rest der ganzzahligen Division
    int width=number%5; // Breite ist von 0 bis 4
    //--- Geben wir die Farbe als Eigenschaft PLOT_LINE_WIDTH
    PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
    //--- Schreiben wir die Linienbreite
    comm=comm+" Width="+IntegerToString(width);

    //--- Block mit Änderungen des Linienstils
    number=MathRand();
    //--- Teiler der Zahl entspricht der Größe des Arrays styles
    int size=ArraySize(styles);
    //--- Erhalten wir den Index als den Rest der ganzzahligen Division, um ein neuer Stil
    int style_index=number%size;
    //--- Geben wir die Farbe als Eigenschaft PLOT_LINE_COLOR an
    PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
    //--- Schreiben wir den Linienstil
    comm=EnumToString(styles[style_index])+", "+comm;
    //--- Zeigen wir Informationen auf dem Chart durch den Kommentar
    Comment(comm);
; }

```

## DRAW\_COLOR\_SECTION

DRAW\_COLOR\_SECTION ist eine farbige version des Stils [DRAW\\_SECTION](#), aber es erlaubt jede Sektion mit einer eigenen Farbe zu zeichnen. Stil DRAW\_COLOR\_SECTION, wie alle farbigen Stile mit **COLOR** in ihren Namen, hat ein besonderer Indikator-Puffer, der den Index (Nummer) der Farbe aus dem angegebenen Farbarray erhielt. So kann die Farbe jeder Sektion gesetzt werden, wenn Sie die Farbindex für die Balken, auf dem das Sektion ende ist, angeben.

Die Breite, Farbe und Stil der Sektionen können wie für [DRAW\\_SECTION](#) durch [Compiler-Direktiven](#) oder dynamisch mit der [PlotIndexSetInteger\(\)](#)-Funktion angegeben werden. Dynamische Veränderungen in den Eigenschaften der graphischen Konstruktion ermöglichen es Ihnen, die Indikatoren zu "wiederbeleben" so, dass sie ihr Aussehen ändern je nach der aktuellen Situation.

Die Sektionen werden aus einem nicht leeren Wert zu einem anderen leeren Wert des Indikator-Puffers gezeichnet, leere Werte werden ignoriert. Um anzugeben, welchen Wert als "leer" betrachtet werden soll, setzen Sie diesen Wert in der Eigenschaft [PLOT\\_EMPTY\\_VALUE](#). Zum Beispiel, wenn der Indikator wie Sektionen auf nicht leere Werte gezeichnet werden soll, müssen Sie einen Nullwert als leer angeben:

```
//--- Der Leerer Wert (0) wird nicht in der Zeichnung teilnehmen  
PlotIndexSetDouble(Index_der_Darstellung_DRAW_COLOR_SECTION, PLOT_EMPTY_VALUE, 0);
```

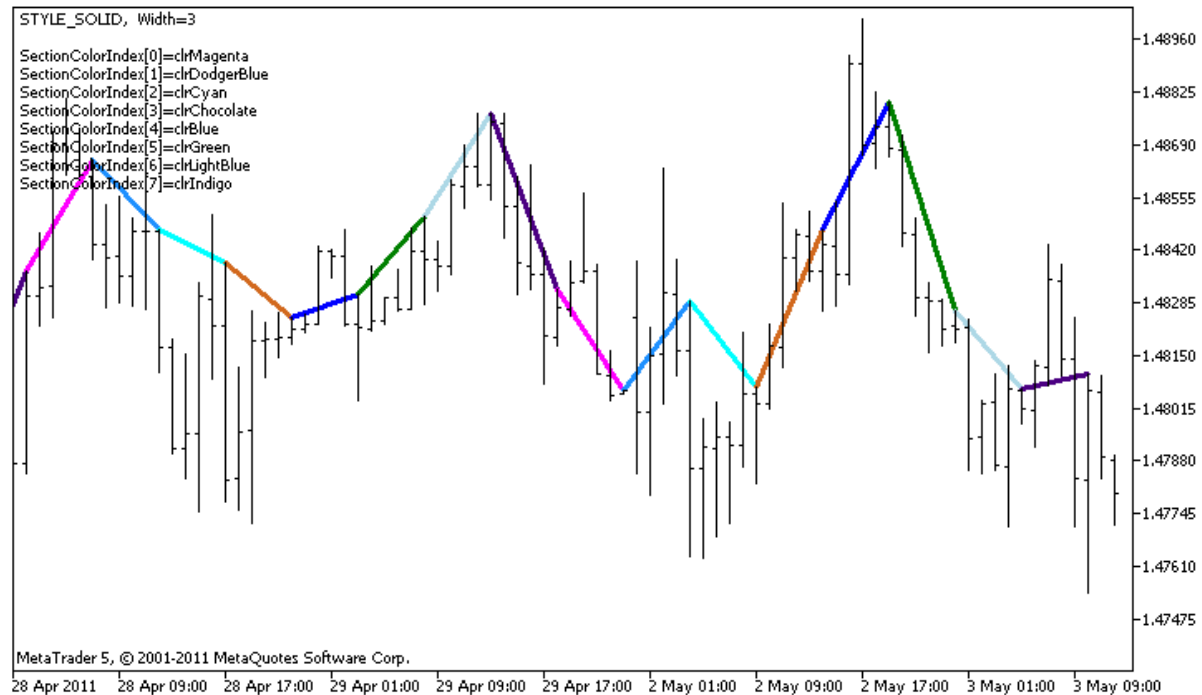
Füllen Sie immer alle Elemente des Indikator-Puffers mit den Werten, geben sie leere Werte für nicht-gezeichnete Elemente.

Die erforderliche Anzahl der Puffer für den Bau von DRAW\_COLOR\_SECTION ist 2.

- einen Puffer für Speicherung die Werte des Indikators, auf denen die Linie gezeichnet wird;
- ein Puffer, um die Farbindex des Segments (es its sinnvoll nur für Nicht-leere Werte) zu speichern.

Farben können mit der Compiler-Direktive [#property indicator\\_color1](#) durch Kommas angegeben werden. Die Anzahl der Farben darf maximal 64 sein.

Ein Beispiel für einen Indikator, der farbige Sektionen von 5 Balken auf die High-Preise zeichnet. Farbe, Breite und Stil der Sektionen werden nach dem Zufallsprinzip jede N Ticks geändert.



Bitte beachten Sie, dass für `plot1` mit dem `DRAW_COLOR_SECTION`-Stil werden 8 Farben mithilfe Compiler-Direktive `#property` angegeben. Dann werden Farben zufällig aus Farbbarray `colors[]` in der Funktion `OnCalculate()` angegeben.

Der Parameter `N` ist in [externen Parametern](#) des Indikators angegeben, damit kann es manuell angegeben werden (die Registerkarte "Einstellungen" im Indikatoreigenschaften-Fenster).

```
//+-----+
//|                                     DRAW_COLOR_SECTION.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Der Indikator demonstriert DRAW_COLOR_SECTION"
#property description "Zeichnet farbige Sektionen von bestimmten Anzahl der Balken"
#property description "Farbe, Breite und Stil der Sektionen werden zufällig"
#property description "jede N Ticks verändert"

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots 1
//--- plot ColorSection
#property indicator_label1 "ColorSection"
#property indicator_type1  DRAW_COLOR_SECTION
//--- Definieren wir 8 Farben um Segmente zu färben (sie sind im speziellen Array gespeichert)
#property indicator_color1  clrRed,clrGold,clrMediumBlue,clrLime,clrMagenta,clrBrown,clrBlack,clrGrey
#property indicator_style1  STYLE_SOLID
```

```

#property indicator_width1 1
//--- Eingabeparameter
input int      N=5;                // Anzahl der Balken zu ändern
input int      bars_in_section=5;  // Länge der Sektionen in Balken
//--- Hilfsvariable, um Ende der Sektionen zu berechnen
int           divider;
int           color_sections;
//--- Buffer für das Rendering
double        ColorSectionBuffer[];
//--- Puffer zu speichern Farben der Liniensegmente auf jedem Balken
double        ColorSectionColors[];
//--- Array, um die Farbe zu speichern, enthält 14 Elemente
color colors[]=
{
    clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
    clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
};
//--- Array für Speicherung der Linienstile
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOTDOT};
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- indicator buffers mapping
    SetIndexBuffer(0,ColorSectionBuffer,INDICATOR_DATA);
    SetIndexBuffer(1,ColorSectionColors,INDICATOR_COLOR_INDEX);
    //--- Der Leerer Wert (0) wird nicht in der Zeichnung teilnehmen
    PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
    //---- Anzahl der Farben um die Sektionen zu färben
    color_sections=8;    // Sehen Sie Kommentar zur Eigenschaft #property indicator_color1
    //--- Überprüfen wir die Indikator-Parameter
    if(bars_in_section<=0)
    {
        PrintFormat("Ungültiger Wert für die Länge der Sektion=%d",bars_in_section);
        return(INIT_PARAMETERS_INCORRECT);
    }
    else divider=color_sections*bars_in_section;
    //---
    return(INIT_SUCCEEDED);
};
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],

```

```

        const double &low[],
        const double &close[],
        const long &tick_volume[],
        const long &volume[],
        const int &spread[])

{
    static int ticks=0;
    //--- Zählen wir Ticks, um den Stil, die Farbe und Breite der Linie zu ändern
    ticks++;
    //--- Wenn eine kritische Anzahl von Ticks angesammelt hat
    if(ticks>=N)
    {
        //--- Ändern wir die Eigenschaften der Linie
        ChangeLineAppearance();
        //--- Ändern wir die Farben der Sektionen
        ChangeColors(colors,color_sections);
        //--- Setzen wir den Zähler der Ticks auf Null
        ticks=0;
    }

    //--- Anzahl der Balken, mit dem die Berechnung des Indikators beginnt
    int start=0;
    //--- Wenn der Indikator bevor berechnet wurde, dann setzen wir start bei dem vorherigen
    if(prev_calculated>0) start=prev_calculated-1;
    //--- alle Berechnungen der Indikatorwerte
    for(int i=start;i<rates_total;i++)
    {
        //--- Wenn der Nummer des Balkens ist teilbar durch Sektion_Länge, bedeutet dies
        if(i%bars_in_section==0)
        {
            //--- Ende der Segment ist auf den High-Preis dieses Balkens gesetzt
            ColorSectionBuffer[i]=high[i];
            //--- Rest der Division von Balkennummern durch Sektion_Länge*Anzahl_der_Farben
            int rest=i%divider;
            //Erhalten wir den Farbenummer = von 0 bis Anzahl_der_Farben-1
            int color_indext=rest/bars_in_section;
            ColorSectionColors[i]=color_indext;
        }
        //--- Wenn der Rest der Division gleich bars ist,
        else
        {
            //--- Wenn etwas nicht passt, dann überspringen wir diesen Balken - setzen 0
            ColorSectionBuffer[i]=0;
        }
    }

    //--- den Wert prev_calculated für den nächsten Anruf der Funktion zurückgeben
    return(rates_total);
}
//+-----+

```

```

//| Ändert die Farbe der Linienteile |
//+-----+
void ChangeColors(color &cols[],int plot_colors)
{
//--- Anzahl der Farben
    int size=ArraySize(cols);
//---
    string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- Für jede Farbeindex definieren wir eine neue Farbe zufällig
    for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
    {
        //--- erhalten wir eine Zufallszahl
        int number=MathRand();
        //--- Erhalten wir den Index in Array col[] als den Rest der ganzzahligen Division
        int i=number%size;
        //--- Geben wir die Farbe für jeden Index als Eigenschaft PLOT_LINE_COLOR an
        PlotIndexSetInteger(0, // Nummer der Grafikstile
                            PLOT_LINE_COLOR, // Property Identifier
                            plot_color_ind, // Farbeindex, in dem wir die Farbe s
                            cols[i]); // neue Farbe

        //--- schreiben wir die Farbe
        comm=comm+StringFormat("SectionColorIndex[%d]=%s \r\n",plot_color_ind,ColorToString(cols[i]));
        ChartSetString(0,CHART_COMMENT,comm);
    }
//---
; }
//+-----+
//| Ändert das Aussehen der angezeigten Linie im Indikator |
//+-----+
void ChangeLineAppearance()
{
//--- String um Informationen über die Eigenschaften der Linie zu erstellen
    string comm="";
//--- Block mit Änderungen der Linienbreite
    int number=MathRand();
//--- Erhalten wir die Breite als den Rest der ganzzahligen Division
    int width=number%5; // Breite ist von 0 bis 4
//--- Geben wir die Farbe als Eigenschaft PLOT_LINE_WIDTH
    PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Schreiben wir die Linienbreite
    comm=comm+" Width="+IntegerToString(width);

//--- Block mit Änderungen des Linienstils
    number=MathRand();
//--- Teiler der Zahl entspricht der Größe des Arrays styles
    int size=ArraySize(styles);
//--- Erhalten wir den Index als den Rest der ganzzahligen Division, um ein neuer Stil
    int style_index=number%size;

```

```
//--- Geben wir die Farbe als Eigenschaft PLOT_LINE_COLOR an
    PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- Schreiben wir den Linienstil
    comm=EnumToString(styles[style_index])+", "+comm;
//--- Zeigen wir Informationen auf dem Chart durch den Kommentar
    Comment(comm);
; }
```

## DRAW\_COLOR\_HISTOGRAM

DRAW\_COLOR\_HISTOGRAM-Stil zeichnet ein Histogramm der Farbsäulen von Null bis den angegebenen Wert. Die Werte werden aus den Indikator-Puffer aufgenommen. Jede Säule kann eine eigene Farbe aus einer vorgegebenen Satz von Farben haben.

Die Breite, Farbe und Stil des Histogramms können wie für [DRAW\\_HISTOGRAM](#) durch [Compiler-Direktiven](#) oder dynamisch mit der [PlotIndexSetInteger\(\)](#)-Funktion angegeben werden. Dynamische Veränderungen in den Eigenschaften der graphischen Konstruktion ermöglichen es Ihnen, den Aussehens des Histogramms je nach der aktuellen Situation zu ändern.

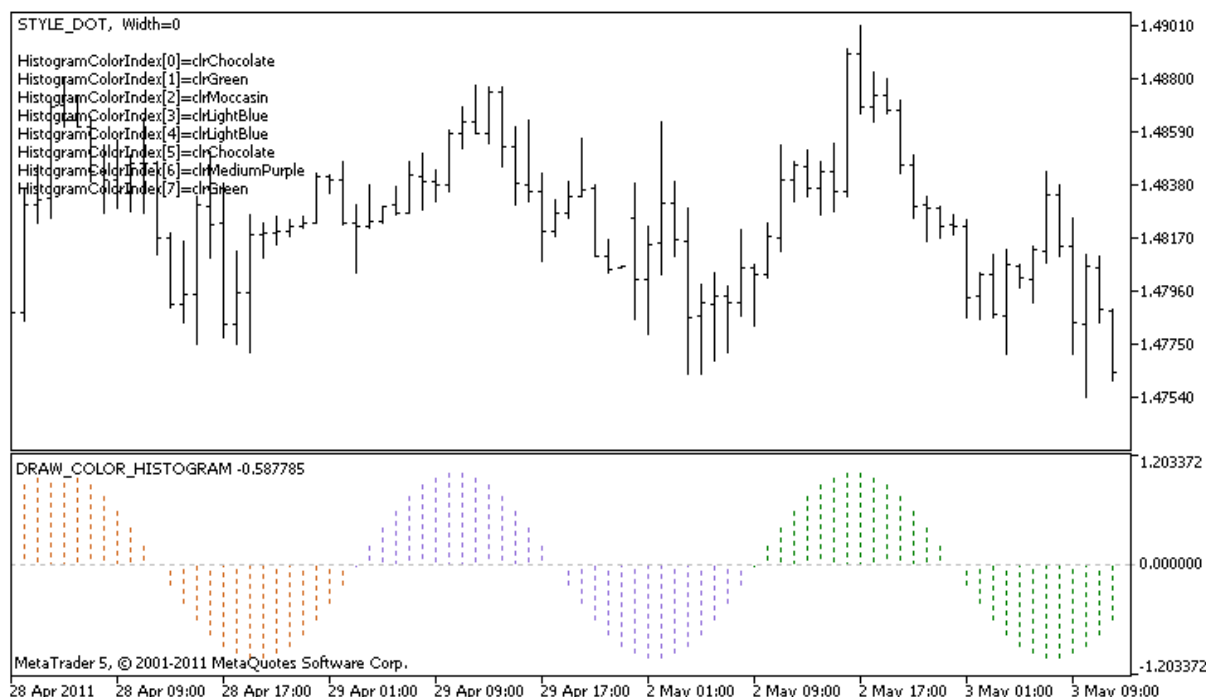
Da wird auf jeden Balken eine Säule von Null-Ebene gezeichnet, dann wird DRAW\_COLOR\_HISTOGRAM besser für Zeichnung in einem separaten Fenster verwendet. Meistens wird diese Art der graphischen Konstruktion an einen Oszillator Art von Indikatoren, z.B. [Awesome Oscillator](#) oder [Market Facilitation Index](#) verwendet. Für leere nicht gezeichnete Werte ist es genug Null-Werte anzugeben.

Die erforderliche Anzahl der Puffer für den Bau von DRAW\_COLOR\_HISTOGRAM ist 2.

- ein Puffer für die Speicherung von Nicht-Null vertikales Segment an jedem Balken, ist das zweite Ende des Segments immer an der Null-Linie des Indikators;
- ein Puffer, um die Farbindex des Segments (es its sinnvoll nur für Nicht-leere Werte) zu speichern.

Farben können mit der Compiler-Direktive #property indicator\_color1 durch Kommas angegeben werden. Die Anzahl der Farben darf maximal 64 sein.

Ein Beispiel für einen Indikator, der eine Sinuskurve der angegebenen Farbe auf Funktion [MathSin\(\)](#) zeichnet. Die Farbe, Breite und Stil **aller** Säule des Histogramms werden nach dem Zufallsprinzip jede N Ticks verändert. Parameter bars gibt den Zeitraum der Sinuskurve, d.h. in einer bestimmten Anzahl von Balken wird die Sinuskurve ihren Zyklus wiederholen.



Bitte beachten Sie, dass für **plot1** mit dem DRAW\_COLOR\_HISTOGRAM-Stil ursprünglich 5 Farben mit Compiler-Direktiven [#property](#) angegeben werden, und dann, in der [OnCalculate\(\)](#)-Funktion, werden



die Farben nach dem Zufallsprinzip aus 14 Farben, die in Array colors[] gespeichert sind, ausgewählt. Der Parameter N wird in [externen Parametern](#) des Indikators angegeben, damit kann es manuell angegeben werden (die Registerkarte "Einstellungen" im Indikatoreigenschaften-Fenster).

```
//+-----+
//|                                     DRAW_COLOR_HISTOGRAM.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Der Indikator demonstriert DRAW_COLOR_HISTOGRAM"
#property description "Zeichnet die Sinuskurve als ein Histogramm in einem separaten Fe
#property description "Farbe und Breite der Säulen werden zufällig"
#property description "jede N Ticks verändert"
#property description "Parameter bars setzt die Anzahl der Balken um die Sinuskurve zu

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots 1
//--- Eingabeparameter
input int      bars=30;           // Zeitraum der Sinuskurve in Balken
input int      N=5;              // Anzahl der Ticks um Histogramm zu ändern
//--- plot Color_Histogram
#property indicator_label1  "Color_Histogram"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
//--- Definieren wir 8 Farben um Segmente zu färben (sie sind im speziellen Array gesp
#property indicator_color1  clrRed,clrGreen,clrBlue,clrYellow,clrMagenta,clrCyan,clrMa
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- Puffer mit Werte
double         Color_HistogramBuffer[];
//--- Puffer mit Farbindizes
double         Color_HistogramColors[];
//--- Faktor für 2Pi Winkel im Bogenmaß durch Multiplikation mit der Parameter bars
double         multipliern;
int            color_sections;
//--- Array, um die Farbe zu speichern, enthält 14 Elemente
color colors[]=
{
    clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
    clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
};
//--- Array für Speicherung der Linienstile
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+-----+
//| Custom indicator initialization function |
```

```

//+-----+
int OnInit()
{
//--- indicator buffers mapping
    SetIndexBuffer(0,Color_HistogramBuffer,INDICATOR_DATA);
    SetIndexBuffer(1,Color_HistogramColors,INDICATOR_COLOR_INDEX);
//---- Anzahl der Farben um die Sinuskurve zu färben
    color_sections=8;    // Sehen Sie Kommentar zur Eigenschaft #property indicator_col
//--- Berechnen wir den Multiplikator
    if(bars>1)multiplier=2.*M_PI/bars;
    else
    {
        PrintFormat("Geben Sie den Wert von bars=%d größer als 1",bars);
        //--- Vorzeitige Beendigung des Indikators
        return(INIT_PARAMETERS_INCORRECT);
    }
//---
    return(INIT_SUCCEEDED);
};

//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{
    static int ticks=0;
//--- Zählen wir Ticks, um den Stil, die Farbe und Breite der Linie zu ändern
    ticks++;
//--- Wenn eine kritische Anzahl von Ticks angesammelt hat
    if(ticks>=N)
    {
        //--- Ändern wir die Eigenschaften der Linie
        ChangeLineAppearance();
        //--- Ändern wir die Farben des Histogramms
        ChangeColors(colors,color_sections);
        //--- Setzen wir den Zähler der Ticks auf Null
        ticks=0;
    }
};

//--- Berechnen der Indikatorwerte
int start=0;

```

```

//--- Wenn die Berechnung auf einem vorherigen Lauf von OnCalculate gemacht war
if(prev_calculated>0) start=prev_calculated-1; // Setzen wir den Beginn der Berechnung
//--- Füllen den Indikator-Puffer mit Werten
for(int i=start;i<rates_total;i++)
{
    //--- Wert
    Color_HistogramBuffer[i]=sin(i*multiplier);
    //--- Farbe
    int color_index=i%(bars*color_sections);
    color_index/=bars;
    Color_HistogramColors[i]=color_index;
};
//--- den Wert prev_calculated für den nächsten Anruf der Funktion zurückgeben
return(rates_total);
};
//+-----+
//| Ändert die Farbe der Linienteile |
//+-----+
void ChangeColors(color &cols[],int plot_colors)
{
    //--- Anzahl der Farben
    int size=ArraySize(cols);
    //---
    string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

    //--- Für jede Farbeindex definieren wir eine neue Farbe zufällig
    for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
    {
        //--- erhalten wir eine Zufallszahl
        int number=MathRand();
        //--- Erhalten wir den Index in Array col[] als den Rest der ganzzahligen Division
        int i=number%size;
        //--- Geben wir die Farbe für jeden Index als Eigenschaft PLOT_LINE_COLOR an
        PlotIndexSetInteger(0, // Nummer der Grafikstile
                           PLOT_LINE_COLOR, // Property Identifier
                           plot_color_ind, // Farbeindex, in dem wir die Farbe setzen
                           cols[i]); // neue Farbe

        //--- schreiben wir die Farbe
        comm=comm+StringFormat("HistogramColorIndex[%d]=%s \r\n",plot_color_ind,ColorToString(cols[i]));
        ChartSetString(0,CHART_COMMENT,comm);
    };
    //---
};
//+-----+
//| Ändert das Aussehen der angezeigten Linie im Indikator |
//+-----+
void ChangeLineAppearance()
{
    //--- String um Informationen über die Eigenschaften der Linie zu erstellen

```

```

    string comm="";
//--- Block mit Änderungen der Linienbreite
    int number=MathRand();
//--- Erhalten wir die Breite als den Rest der ganzzahligen Division
    int width=number%5; // Breite ist von 0 bis 4
//--- Geben wir die Farbe als Eigenschaft PLOT_LINE_WIDTH
    PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Schreiben wir die Linienbreite
    comm=comm+" Width="+IntegerToString(width);

//--- Block mit Änderungen des Linienstils
    number=MathRand();
//--- Teiler der Zahl entspricht der Größe des Arrays styles
    int size=ArraySize(styles);
//--- Erhalten wir den Index als den Rest der ganzzahligen Division, um ein neuer Stil
    int style_index=number%size;
//--- Geben wir die Farbe als Eigenschaft PLOT_LINE_COLOR an
    PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- Schreiben wir den Linienstil
    comm=EnumToString(styles[style_index])+", "+comm;
//--- Zeigen wir Informationen auf dem Chart durch den Kommentar
    Comment(comm);
; }

```

## DRAW\_COLOR\_HISTOGRAM2

DRAW\_COLOR\_HISTOGRAM2-Stil zieht ein Histogramm der angegebenen Farbe - die vertikalen Segmente auf die Werte der zwei Indikator-Puffer. Aber im Gegensatz zu den einfarbigen DRAW\_HISTOGRAM2, in diesem Stil zu jedem Histogrammsäule können Sie ihre eigene Farbe aus einer vordefinierten Gruppe angeben. Die Werte der Segmentende werden aus den Indikator-Puffer aufgenommen.

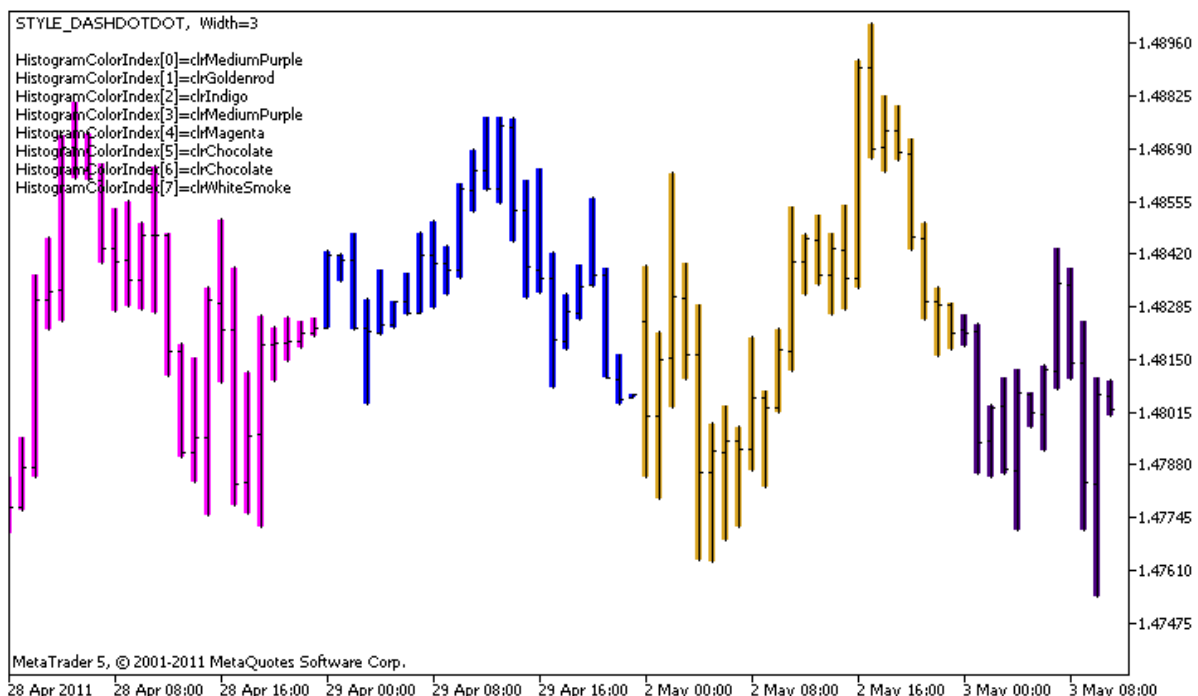
Breite, Stil und Farbe des Histogramms können wie für [DRAW\\_HISTOGRAM2](#) durch [Compiler-Direktiven](#) oder dynamisch mit der [PlotIndexSetInteger\(\)](#)-Funktion angegeben werden. Dynamische Veränderungen in den Eigenschaften der graphischen Konstruktion ermöglichen es Ihnen, den Aussehens des Histogramms je nach der aktuellen Situation zu ändern.

DRAW\_COLOR\_HISTOGRAM2-Stil kann in einem separaten Grafikfenster und im Hauptfenster verwendet werden. Leere Werte werden nicht gezeichnet, alle Werte in Indikator-Puffern müssen explizit angegeben werden. Puffer werden nicht mit einem leeren Wert initialisiert.

Die erforderliche Anzahl der Puffer für den Bau von DRAW\_COLOR\_HISTOGRAM2 ist 3:

- zwei Puffer für Speicherung der oberen und unteren Enden des vertikalen Segments auf jeden Balken;
- ein Puffer, um die Farbindex des Segments (es ist sinnvoll nur für Nicht-leere Werte) zu speichern.

Ein Beispiel für einen Indikator, der ein Histogramm der angegebenen Farbe zwischen den Preisen von High und Low zeichnet. Für jeden Tag der Woche sind die Histogrammlinien mit ihrer Farbe gezeichnet. Die Farbe jeden Tages, Breite und Stil des Histogramms werden nach dem Zufallsprinzip jede N Ticks verändert.



Bitte beachten Sie, dass für `plot1` mit dem DRAW\_COLOR\_HISTOGRAM2-Stil ursprünglich 5 Farben mit Compiler-Direktiven [#property](#) angegeben werden, und dann, in der [OnCalculate\(\)](#)-Funktion, werden die Farben nach dem Zufallsprinzip aus 14 Farben, die in `Array colors[]` gespeichert sind, ausgewählt.

Der Parameter N ist in [externen Parametern](#) des Indikators angegeben, damit kann es manuell angegeben werden (die Registerkarte "Einstellungen" im Indikatoreigenschaften-Fenster).

```
//+-----+
//|                                     DRAW_COLOR_HISTOGRAM2.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Der Indikator demonstriert DRAW_COLOR_HISTOGRAM2"
#property description "Zeichnet ein Segment auf jedem Balken zwischen Open und Close"
#property description "Farbe, Breite und Stil werden zufällig"
#property description "jede N Ticks verändert"

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots 1
//--- plot ColorHistogram_2
#property indicator_label1  "ColorHistogram_2"
#property indicator_type1   DRAW_COLOR_HISTOGRAM2
//--- Definieren wir 5 Farben um Histogramm nach Wochentag zu färben (sie sind im spez
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1

//--- Eingabeparameter
input int      N=5;           // Anzahl der Ticks um Histogramm zu ändern
int        color_sections;
//--- Puffer mit Werte
double     ColorHistogram_2Buffer1[];
double     ColorHistogram_2Buffer2[];
//--- Puffer mit Farbindizes
double     ColorHistogram_2Colors[];
//--- Array, um die Farbe zu speichern, enthält 14 Elemente
color colors[]=
{
    clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
    clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
};
//--- Array für Speicherung der Linienstile
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+-----+
//| Custom indicator initialization function |
//+-----+

int OnInit()
{

```

```

//--- indicator buffers mapping
SetIndexBuffer(0,ColorHistogram_2Buffer1,INDICATOR_DATA);
SetIndexBuffer(1,ColorHistogram_2Buffer2,INDICATOR_DATA);
SetIndexBuffer(2,ColorHistogram_2Colors,INDICATOR_COLOR_INDEX);
//--- Leerer Wert setzen
PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---- Anzahl der Farben um die Sinuskurve zu färben
color_sections=8; // Sehen Sie Kommentar zur Eigenschaft #property indicator_col
//---
return(INIT_SUCCEEDED);
;
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
    static int ticks=0;
//--- Zählen wir Ticks, um den Stil, die Farbe und Breite der Linie zu ändern
    ticks++;
//--- Wenn eine kritische Anzahl von Ticks angesammelt hat
    if(ticks>=N)
    {
        //--- Ändern wir die Eigenschaften der Linie
        ChangeLineAppearance();
//--- Ändern wir die Farben des Histogramms
        ChangeColors(colors,color_sections);
//--- Setzen wir den Zähler der Ticks auf Null
        ticks=0;
    }
;

//--- Berechnen der Indikatorwerte
    int start=0;
//--- Um den Wochentag durch den Öffnungszeiten jedes Balkens
    MqlDateTime dt;
//--- Wenn die Berechnung auf einem vorherigen Lauf von OnCalculate gemacht war
    if(prev_calculated>0) start=prev_calculated-1; // Setzen wir den Beginn der Berechnung
//--- Füllen den Indikator-Puffer mit Werte
    for(int i=start;i<rates_total;i++)
    {
        TimeToStruct(time[i],dt);
    }
}

```

```

    //--- Wert
    ColorHistogram_2Buffer1[i]=high[i];
    ColorHistogram_2Buffer2[i]=low[i];
    //--- Angeben wir den Farbindex nach Wochentag
    int day=dt.day_of_week;
    ColorHistogram_2Colors[i]=day;

    ;}

//--- den Wert prev_calculated für den nächsten Anruf der Funktion zurückgeben
    return(rates_total);

    ;}

//+-----+
//| Ändert die Farbe der Linieteile |
//+-----+
void ChangeColors(color &cols[],int plot_colors)
{
//--- Anzahl der Farben
    int size=ArraySize(cols);
//---
    string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- Für jede Farbeindex definieren wir eine neue Farbe zufällig
    for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
    {
        //--- erhalten wir eine Zufallszahl
        int number=MathRand();
        //--- Erhalten wir den Index in Array col[] als den Rest der ganzzahligen Division
        int i=number%size;
        //--- Geben wir die Farbe für jeden Index als Eigenschaft PLOT_LINE_COLOR an
        PlotIndexSetInteger(0, // Nummer der Grafikstile
                            PLOT_LINE_COLOR, // Property Identifier
                            plot_color_ind, // Farbindex, in dem wir die Farbe setzen
                            cols[i]); // neue Farbe

        //--- schreiben wir die Farbe
        comm=comm+StringFormat("HistogramColorIndex[%d]=%s \r\n",plot_color_ind,ColorToString(cols[i]));
        ChartSetString(0,CHART_COMMENT,comm);

    ;}

//---
    ;}

//+-----+
//| Ändert das Aussehen der angezeigten Linie im Indikator |
//+-----+
void ChangeLineAppearance()
{
//--- String um Informationen über die Eigenschaften der Linie zu erstellen
    string comm="";
//--- Block mit Änderungen der Linienbreite
    int number=MathRand();
//--- Erhalten wir die Breite als den Rest der ganzzahligen Division
    int width=number%5; // Breite ist von 0 bis 4

```



```
//--- Geben wir die Farbe als Eigenschaft PLOT_LINE_WIDTH
    PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Schreiben wir die Linienbreite
    comm=comm+" Width="+IntegerToString(width);

//--- Block mit Änderungen des Linienstils
    number=MathRand();
//--- Teiler der Zahl entspricht der Größe des Arrays styles
    int size=ArraySize(styles);
//--- Erhalten wir den Index als den Rest der ganzzahligen Division, um ein neuer Stil
    int style_index=number%size;
//--- Geben wir die Farbe als Eigenschaft PLOT_LINE_COLOR an
    PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- Schreiben wir den Linienstil
    comm=EnumToString(styles[style_index])+", "+comm;
//--- Zeigen wir Informationen auf dem Chart durch den Kommentar
    Comment(comm);
; }
```

## DRAW\_COLOR\_ARROW

Der DRAW\_COLOR\_ARROW-Stil zeichnet Pfeile (Zeichen aus dem [Wingdings](#)-Satz) der Farbe auf den Wert des Indikator-Puffers. Im Gegensatz zum Stil DRAW\_ARROW, ist es möglich, die Farbe für jede Zeichen aus einer vordefinierten Gruppe von Farben, die durch die Eigenschaft [indicator\\_color1](#) definiert werden, einzustellen.

Die Breite und die Farbe der Symbole können wie für [DRAW\\_ARROW](#) durch [Compiler-Direktiven](#) oder dynamisch mit der [PlotIndexSetInteger\(\)](#)-Funktion angegeben werden. Dynamische Veränderungen in den Eigenschaften der graphischen Konstruktion ermöglichen es Ihnen, den Aussehens eines Indikators zu ändern je nach der aktuellen Situation.

Der Symbolcode wird mit der [PLOT\\_ARROW](#)-Eigenschaft eingegeben.

```
//--- Den Symbolcode aus Wingdings-Satz für PLOT_ARROW definieren
PlotIndexSetInteger(0, PLOT_ARROW, code);
```

Der Standardwert ist PLOT\_ARROW=159 (Kreis).

Jeder Pfeil ist eigentlich ein Zeichen, dass die Höhe und den Ankerpunkt hat, und kann einige wichtige Information über den Chart (z. B. der Schlusskurs am Balken) bedecken. Daher können Sie optional den vertikalen Verschiebung in Pixel, die nicht auf der Skala des Charts abhängig ist, angeben. Bei dieser Anzahl von Pixeln werden die Pfeile visuell vertikal verschoben, obwohl der Wert des Indikators gleich bleiben wird:

```
//--- Definieren wir vertikale Verschiebung der Pfeile in Pixel
PlotIndexSetInteger(0, PLOT_ARROW_SHIFT, shift);
```

Ein negativer Wert von PLOT\_ARROW\_SHIFT bedeutet Verschiebung des Pfeils nach oben, ein positiver Wert verschiebt den Pfeil nach unten.

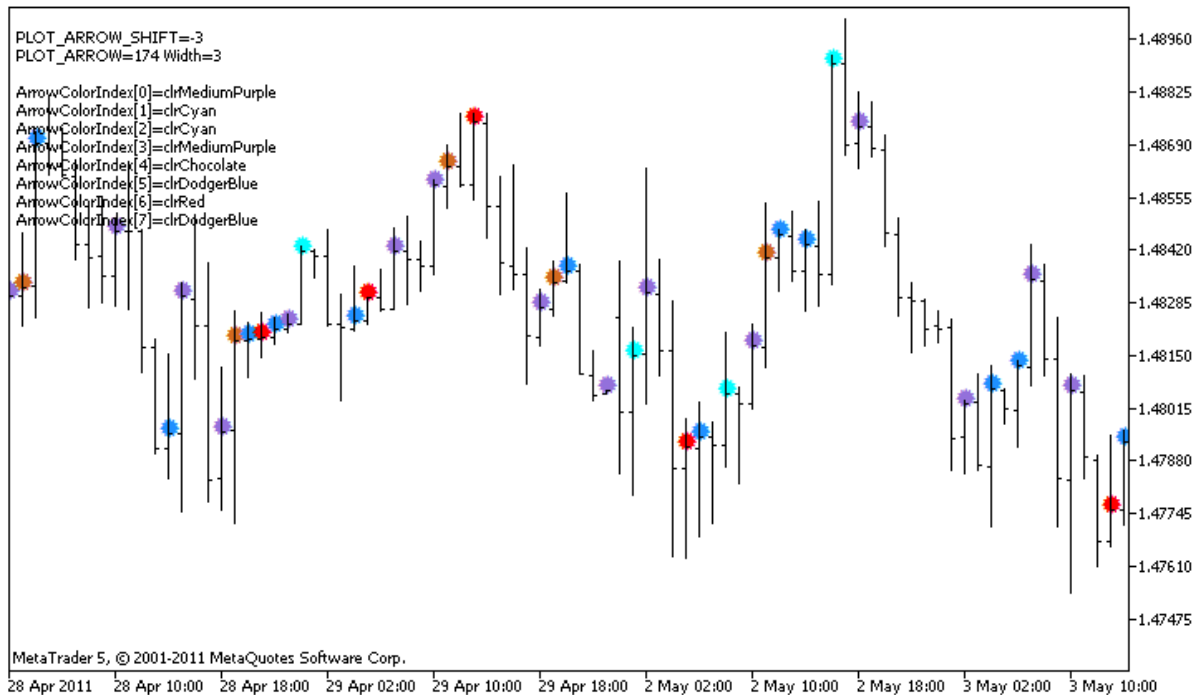
DRAW\_COLOR\_ARROW-Stil kann in einem separaten Grafikfenster und im Hauptfenster verwendet werden. Leere Werte werden nicht gezeichnet und nicht in den "Data Window" angezeigt, alle Werte in Indikator-Puffern müssen explizit angegeben werden. Puffer werden nicht mit einem leeren Wert initialisiert.

```
//--- Leerer Wert setzen
PlotIndexSetDouble(Index_der_Darstellung_DRAW_COLOR_ARROW, PLOT_EMPTY_VALUE, 0);
```

Die erforderliche Anzahl der Puffer für den Bau von DRAW\_COLOR\_ARROW ist 4.

- einen Puffer, um den Wert der Preis, auf den das Symbol gezeichnet wird (plus Verschiebung in Pixel, die in PLOT\_ARROW\_SHIFT-Eigenschaft angegeben wird), zu speichern;
- einen Puffer, um die Farbindex des Pfeils (es ist sinnvoll nur für Nicht-leere Werte) zu speichern.

Ein Beispiel des Indikators, der Pfeile auf jeder Balken, dessen Schlusspreis Close mehr als Schlusspreis des vorherigen Balkens ist, zeichnet. Die Breite, Verschiebung und Zeichencode aller Pfeile werden nach dem Zufallsprinzip jede N Ticks verändert. Die Symbolfarbe hängt von der Nummer des Balkens, auf der es gezeichnet wird, ab.



In diesem Beispiel, für **plot1** mit dem **DRAW\_COLOR\_ARROW**-Stil werden Farbe und Größe mithilfe Compiler-Direktive [#property](#) angegeben, und dann in der Funktion [OnCalculate\(\)](#) werden Eigenschaften nach dem Zufallsprinzip angegeben. Der Parameter N wird in [externen Parametern](#) des Indikators angegeben, damit kann es manuell angegeben werden (die Registerkarte "Einstellungen" im Indikatoreigenschaften-Fenster).

Bitte beachten Sie, dass ursprünglich acht Farben mit Compiler-Direktiven [#property](#) angegeben werden, und dann, in der [OnCalculate\(\)](#)-Funktion, wird die Farbe nach dem Zufallsprinzip aus 14 Farben, die in Array `colors[]` gespeichert sind, ausgewählt.

```
//+-----+
//|                                     DRAW_COLOR_ARROW.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Der Indikator demonstriert DRAW_COLOR_ARROW"
#property description "Auf dem Chart zeichnet es verschiedenfarbige Pfeile, die durch"
#property description "Die Farbe, Größe, Verschiebung und Zeichencode des Pfeils werde"
#property description "jede N Ticks verändert."
#property description "Parameter code gibt den Basiswert an: Code=159 (Kreis)"

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots 1
//--- plot ColorArrow
#property indicator_label1 "ColorArrow"
```

```

#property indicator_type1   DRAW_COLOR_ARROW
//--- Definieren wir 8 Farben um Histogramm nach Wochentag zu färben (sie sind im spez
#property indicator_color1  clrRed,clrBlue,clrSeaGreen,clrGold,clrDarkOrange,clrMagent
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1

//--- Eingabeparameter
input int      N=5;          // Anzahl der Ticks für Änderung
input ushort   code=159;     // Zeichencode zu zeichnen in DRAW_ARROW
int           color_sections;
//--- Indicator-Puffer
double         ColorArrowBuffer[];
//--- Puffer, um die Indexe der Farbe zu speichern
double         ColorArrowColors[];
//--- Array, um die Farbe zu speichern, enthält 14 Elemente
color colors[]=
{
    clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
    clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
};
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- indicator buffers mapping
    SetIndexBuffer(0,ColorArrowBuffer,INDICATOR_DATA);
    SetIndexBuffer(1,ColorArrowColors,INDICATOR_COLOR_INDEX);
    //--- Geben wir Zeichencode um in PLOT_ARROW zu zeichnen an
    PlotIndexSetInteger(0,PLOT_ARROW,code);
    //--- Definieren wir vertikale Verschiebung der Pfeile in Pixel
    PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,5);
    //--- Setzen wir 0 als ein leerer Wert
    PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
    //---- Anzahl der Farben um die Sinuskurve zu färben
    color_sections=8;    // Sehen Sie Kommentar zur Eigenschaft #property indicator_col
    //---
    return(INIT_SUCCEEDED);
};
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],

```

```

        const long &tick_volume[],
        const long &volume[],
        const int &spread[])

{
    static int ticks=0;
    //--- Zählen wir Ticks, um die Farbe, Größe, Verschiebung und Code des Pfeils zu ändern
    ticks++;
    //--- Wenn eine kritische Anzahl von Ticks angesammelt hat
    if(ticks>=N)
    {
        //--- Ändern wir die Eigenschaften der Pfeile
        ChangeLineAppearance();
        //--- Ändern wir die Farben des Histogramms
        ChangeColors(colors,color_sections);
        //--- Setzen wir den Zähler der Ticks auf Null
        ticks=0;
    };

    //--- Block mit Berechnung der Indikatorwerte
    int start=1;
    if(prev_calculated>0) start=prev_calculated-1;
    //--- Berechnungszyklus
    for(int i=1;i<rates_total;i++)
    {
        //--- Wenn der aktuelle Close-Preis ist höher als der vorherige Close-price, dann setzen
        if(close[i]>close[i-1])
            ColorArrowBuffer[i]=close[i];
        //---Ansonsten geben einen leeren Wert an
        else
            ColorArrowBuffer[i]=0;
        //--- Pfeilfarbe
        int index=i%color_sections;
        ColorArrowColors[i]=index;
    };

    //--- return value of prev_calculated for next call
    return(rates_total);
};

//+-----+
//| Ändert die Farbe der Linienteile |
//+-----+
void ChangeColors(color &cols[],int plot_colors)
{
    //--- Anzahl der Farben
    int size=ArraySize(cols);
    //---
    string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

    //--- Für jede Farbeindex definieren wir eine neue Farbe zufällig
    for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)

```

```

{
    //--- erhalten wir eine Zufallszahl
    int number=MathRand();
    //--- Erhalten wir den Index in Array col[] als den Rest der ganzzahligen Division
    int i=number%size;
    //--- Geben wir die Farbe für jeden Index als Eigenschaft PLOT_LINE_COLOR an
    PlotIndexSetInteger(0, // Nummer der Grafikstile
        PLOT_LINE_COLOR, // Property Identifier
        plot_color_ind, // Farbindex, in dem wir die Farbe setzen
        cols[i]); // neue Farbe

    //--- schreiben wir die Farbe
    comm=comm+StringFormat("ArrowColorIndex[%d]=%s \r\n",plot_color_ind,ColorToString(plot_color_ind));
    ChartSetString(0,CHART_COMMENT,comm);
};
//---
; }
//+-----+
//| Ändert das Aussehen der angezeigten Linie im Indikator |
//+-----+
void ChangeLineAppearance()
{
    //--- String um Informationen über die Eigenschaften der Linie zu erstellen
    string comm="";
    //--- Block mit Änderungen der Linienbreite
    int number=MathRand();
    //--- Erhalten wir die Breite als den Rest der ganzzahligen Division
    int width=number%5; // Breite ist von 0 bis 4
    //--- Geben wir die Farbe als Eigenschaft PLOT_LINE_WIDTH
    PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
    //--- Schreiben wir die Linienbreite
    comm=comm+" Width="+IntegerToString(width);

    //--- Block mit Änderungen von Pfeilcode (PLOT_ARROW)
    number=MathRand();
    //--- Erhalten wir den Rest aus Integer-Division, um den neuen Pfeilcode zu berechnen
    int code_add=number%20;
    //--- Setzen wir einen neuen Zeichencode als die Summe von code+code_add
    PlotIndexSetInteger(0,PLOT_ARROW,code+code_add);
    //--- Schreiben wir den Zeichencode PLOT_ARROW
    comm="\r\n"+"PLOT_ARROW="+IntegerToString(code+code_add)+comm;

    //--- Block mit vertikaler Verschiebung der Pfeile in Pixel
    number=MathRand();
    //--- Erhalten wir die Verschiebung als den Rest der ganzzahligen Division
    int shift=20-number%41;
    //--- Einstellen wir eine neue Verschiebung von
    PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
    //--- Schreiben wir die Verschiebung PLOT_ARROW_SHIFT
    comm="\r\n"+"PLOT_ARROW_SHIFT="+IntegerToString(shift)+comm;
}

```

```
//--- Zeigen wir Informationen auf dem Chart durch den Kommentar  
    Comment(comm);  
; }
```

## DRAW\_COLOR\_ZIGZAG

Stil DRAW\_COLOR\_ZIGZAG zeichnet Segmente in verschiedenen Farben auf die Werten der zwei Indikator-Puffer. Dieser Stil ist eine farbige Version des Stils [DRAW\\_ZIGZAG](#) und ermöglicht Ihnen, die Farbe für jedes Segment aus einer vorgegebenen Satz von Farben anzugeben. Die Segmente werden vom dem Wert im ersten Puffer bis dem Wert im zweiten Indikator-Puffer gezeichnet. Keiner der Puffer darf nicht nur leere Werte erhalten, da in diesem Fall nichts gezeichnet wird.

Die Breite, Farbe und Stil der Linie können wie für [DRAW\\_ZIGZAG](#) durch [Compiler-Direktiven](#) oder dynamisch mit der [PlotIndexSetInteger\(\)](#)-Funktion angegeben werden. Dynamische Veränderungen in den Eigenschaften der graphischen Konstruktion ermöglichen es Ihnen, die Indikatoren zu "wiederbeleben" so, dass sie ihr Aussehen ändern je nach der aktuellen Situation.

Die Segmente werden vom nicht leeren Wert eines Puffers zum nicht leeren Wert anderes Indikator-Puffers gezeichnet. Um anzugeben, welchen Wert als "leer" betrachtet werden soll, setzen Sie diesen Wert in der Eigenschaft [PLOT\\_EMPTY\\_VALUE](#):

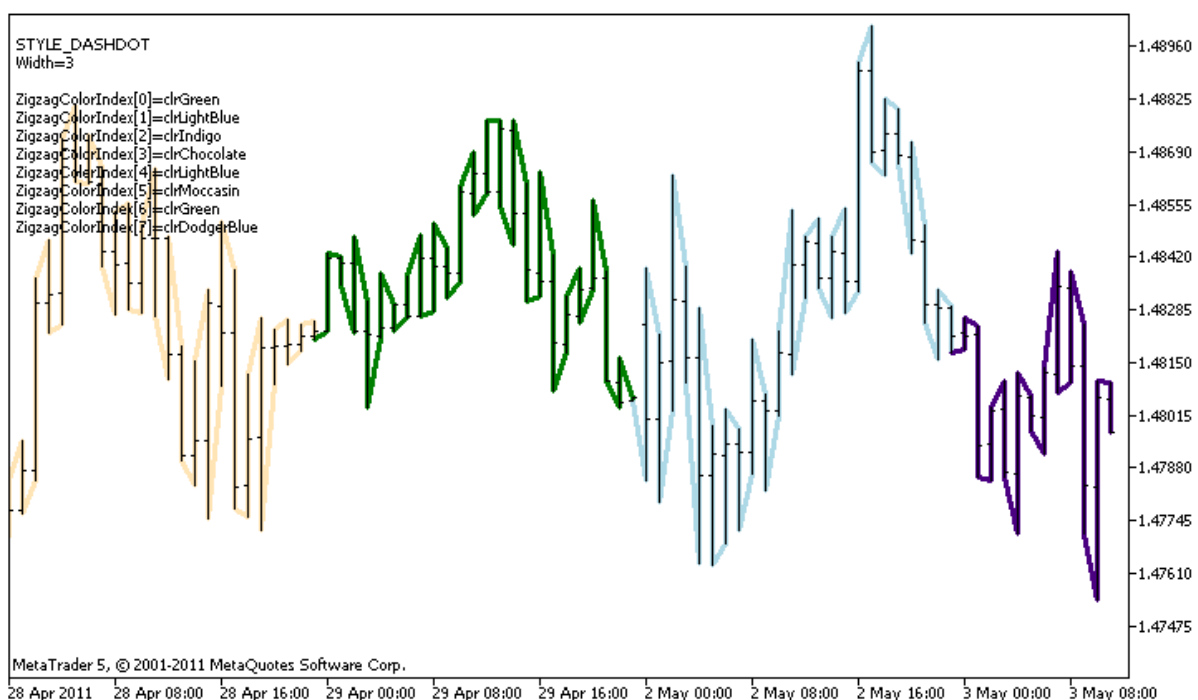
```
//--- Der Leerer Wert (0) wird nicht in der Zeichnung teilnehmen
PlotIndexSetDouble(Index_der_Darstellung_DRAW_COLOR_ZIGZAG, PLOT_EMPTY_VALUE, 0);
```

Immer füllen Sie die Indikator-Puffer mit Werten expliziert aus, geben Sie den leeren Wert in Puffer der Balken zu überspringen.

Die erforderliche Anzahl der Puffer für den Bau von DRAW\_COLOR\_ZIGZAG ist 3:

- zwei Puffer zum Speichern der Werte von Ende der Segmente von Zigiag;
- ein Puffer, um die Farbindex des Segments (es its sinnvoll nur für Nicht-leere Werte) zu speichern.

Ein Beispiel für einen Indikator, der eine Säge auf die High und Low Pries zeichnet. Farbe, Breite und Stil der Linien von Zigzag werden nach dem Zufallsprinzip jede N Ticks geändert.



Bitte beachten Sie, dass für plot1 mit dem DRAW\_COLOR\_ZIGZAG-Stil ursprünglich acht Farben mit



Compiler-Direktiven [#property](#) angegeben werden, und dann, in der [OnCalculate\(\)](#)-Funktion, wird die Farbe nach dem Zufallsprinzip aus 14 Farben, die in `Array colors[]` gespeichert sind, ausgewählt.

Der Parameter N ist in [externen Parametern](#) des Indikators angegeben, damit kann es manuell angegeben werden (die Registerkarte "Einstellungen" im Indikatoreigenschaften-Fenster).

```
//+-----+
//|                                     DRAW_COLOR_ZIGZAG.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Der Indikator demonstriert DRAW_COLOR_ZIGZAG"
#property description "Zeichnet eine gebrochene Linie von farbigen Segmenten, die Farbe"
#property description "Farbe, Breite und Stil der Segmente variieren zufällig"
#property description "jede N Ticks"

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots 1
//--- plot Color_Zigzag
#property indicator_label1 "Color_Zigzag"
#property indicator_type1  DRAW_COLOR_ZIGZAG
//--- Definieren wir 8 Farben um Segmente zu färben (sie sind im speziellen Array gespeichert)
#property indicator_color1 clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrLightBlue,clrDarkBlue
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//--- Eingabeparameter
input int      N=5;           // Anzahl der Ticks für Änderung
int        color_sections;
//--- Puffer mit den Werten der Segmentenden
double     Color_ZigzagBuffer1[];
double     Color_ZigzagBuffer2[];
//--- Puffer mit den Farbindizes der Segmentenden
double     Color_ZigzagColors[];
//--- Array, um die Farbe zu speichern, enthält 14 Elemente
color colors[]=
{
    clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
    clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurple;
};
//--- Array für Speicherung der Linienstile
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOTDOT};
//+-----+
//| Custom indicator initialization function |
//+-----+
```

```

int OnInit()
{
//--- indicator buffers mapping
    SetIndexBuffer(0,Color_ZigzagBuffer1,INDICATOR_DATA);
    SetIndexBuffer(1,Color_ZigzagBuffer2,INDICATOR_DATA);
    SetIndexBuffer(2,Color_ZigzagColors,INDICATOR_COLOR_INDEX);
//---- Anzahl der Farben um Zigzag zu färben
    color_sections=8;    // Sehen Sie Kommentar zur Eigenschaft #property indicator_col
//---
    return(INIT_SUCCEEDED);
};
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{
    static int ticks=0;
//--- Zählen wir Ticks, um den Stil, die Farbe und Breite der Linie zu ändern
    ticks++;
//--- Wenn eine ausreichende Anzahl von Ticks angesammelt hat
    if(ticks>=N)
    {
        //--- Ändern wir die Eigenschaften der Linie
        ChangeLineAppearance();
        //--- Ändern wir die Farben der Sektionen
        ChangeColors(colors,color_sections);
        //--- Setzen wir den Zähler der Ticks auf Null
        ticks=0;
    };

//--- Die Zeitstruktur ist benötigt, um den Wochentag jedes Balkens zu erhalten
    MqlDateTime dt;

//--- Startposition der Berechnungen
    int start=0;
//--- Wenn der Indikator wurde auf dem vorherigen Tick berechnet, dann beginnen wir Be
    if(prev_calculated!=0) start=prev_calculated-1;
//--- Berechnungszyklus
    for(int i=start;i<rates_total;i++)
    {

```

```

//--- Schreiben die Balkenöffnungszeit in die Struktur
TimeToStruct(time[i],dt);

//--- Wenn die balkennummer gerade ist
if(i%2==0)
{
    //--- Schreiben wir High in den ersten Puffer und Low in den Zweiten Puffer
    Color_ZigzagBuffer1[i]=high[i];
    Color_ZigzagBuffer2[i]=low[i];
    //--- Farbe des Segments
    Color_ZigzagColors[i]=dt.day_of_year%color_sections;
};
//--- Balkennummer ist ungerade
else
{
    //--- Füllen den Balken in der umgekehrten Reihenfolge
    Color_ZigzagBuffer1[i]=low[i];
    Color_ZigzagBuffer2[i]=high[i];
    //--- Farbe des Segments
    Color_ZigzagColors[i]=dt.day_of_year%color_sections;
};
};

//--- return value of prev_calculated for next call
return(rates_total);
};

//+-----+
//| Ändert die Farbe der Segmente des Zick-Zacks |
//+-----+
void ChangeColors(color &cols[],int plot_colors)
{
    //--- Anzahl der Farben
    int size=ArraySize(cols);
    //---
    string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

    //--- Für jede Farbeindex definieren wir eine neue Farbe zufällig
    for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
    {
        //--- erhalten wir eine Zufallszahl
        int number=MathRand();
        //--- Erhalten wir den Index in Array col[] als den Rest der ganzzahligen Divisi
        int i=number%size;
        //--- Geben wir die Farbe für jeden Index als Eigenschaft PLOT_LINE_COLOR an
        PlotIndexSetInteger(0, // Nummer der Grafikstile
                           PLOT_LINE_COLOR, // Property Identifizier
                           plot_color_ind, // Farbindex, in dem wir die Farbe s
                           cols[i]); // neue Farbe
        //--- schreiben wir die Farbe
        comm=comm+StringFormat("ZigzagColorIndex[%d]=%s \r\n",plot_color_ind,ColorToStr

```

```

        ChartSetString(0, CHART_COMMENT, comm);
    };
//---
};
//+-----+
//| Ändert das Aussehen der Segmente in Zigzag |
//+-----+
void ChangeLineAppearance()
{
//--- String um Informationen über die Eigenschaften von Color_ZigZag zu erstellen
    string comm="";
//--- Block mit Änderungen der Linienbreite
    int number=MathRand();
//--- Erhalten wir die Breite als den Rest der ganzzahligen Division
    int width=number%5;    // Breite ist von 0 bis 4
//--- Geben wir die Farbe als Eigenschaft PLOT_LINE_WIDTH
    PlotIndexSetInteger(0, PLOT_LINE_WIDTH, width);
//--- Schreiben wir die Linienbreite
    comm=comm+"\r\nWidth="+IntegerToString(width);

//--- Block mit Änderungen des Linienstils
    number=MathRand();
//--- Teiler der Zahl entspricht der Größe des Arrays styles
    int size=ArraySize(styles);
//--- Erhalten wir den Index als den Rest der ganzzahligen Division, um ein neuer Stil
    int style_index=number%size;
//--- Geben wir die Farbe als Eigenschaft PLOT_LINE_COLOR an
    PlotIndexSetInteger(0, PLOT_LINE_STYLE, styles[style_index]);
//--- Schreiben wir den Linienstil
    comm="\r\n"+EnumToString(styles[style_index])+" "+comm;
//--- Zeigen wir Informationen auf dem Chart durch den Kommentar
    Comment(comm);
};

```

## DRAW\_COLOR\_BARS

Stil DRAW\_COLOR\_BARS zieht Balken auf die Werte der vier Indikator-Puffer, die Preise Open, High, Low und Close erhalten. Dieser Stil ist eine erweiterte Version des Stils [DRAW\\_BARS](#) und ermöglicht Ihnen, die Farbe für jeden Balken aus einer vorgegebenen Satz von Farben anzugeben. Es erlaubt individuelle Indikatoren in Form von Balken, darunter ein in separates Grafikfenster und andere Finanzinstrumente, zu erstellen.

Balkenfarbe kann durch [Compiler-Direktiven](#) oder dynamisch mit der [PlotIndexSetInteger\(\)](#)-Funktion angegeben werden. Dynamische Veränderungen in den Eigenschaften der graphischen Konstruktion ermöglichen es Ihnen, die Indikatoren zu "wiederbeleben" so, dass sie ihr Aussehen ändern je nach der aktuellen Situation.

Der Indikator ist nur für die Balken, für die nicht leer Werte **aller** vier Indikator-Puffern angegeben wird, gezeichnet. Um anzugeben, welchen Wert als "leer" betrachtet werden soll, setzen Sie diesen Wert in der Eigenschaft [PLOT\\_EMPTY\\_VALUE](#):

```
//--- Der Leerer Wert (0) wird nicht in der Zeichnung teilnehmen  
PlotIndexSetDouble(Index_der_Darstellung_DRAW_COLOR_BARS,PLOT_EMPTY_VALUE,0);
```

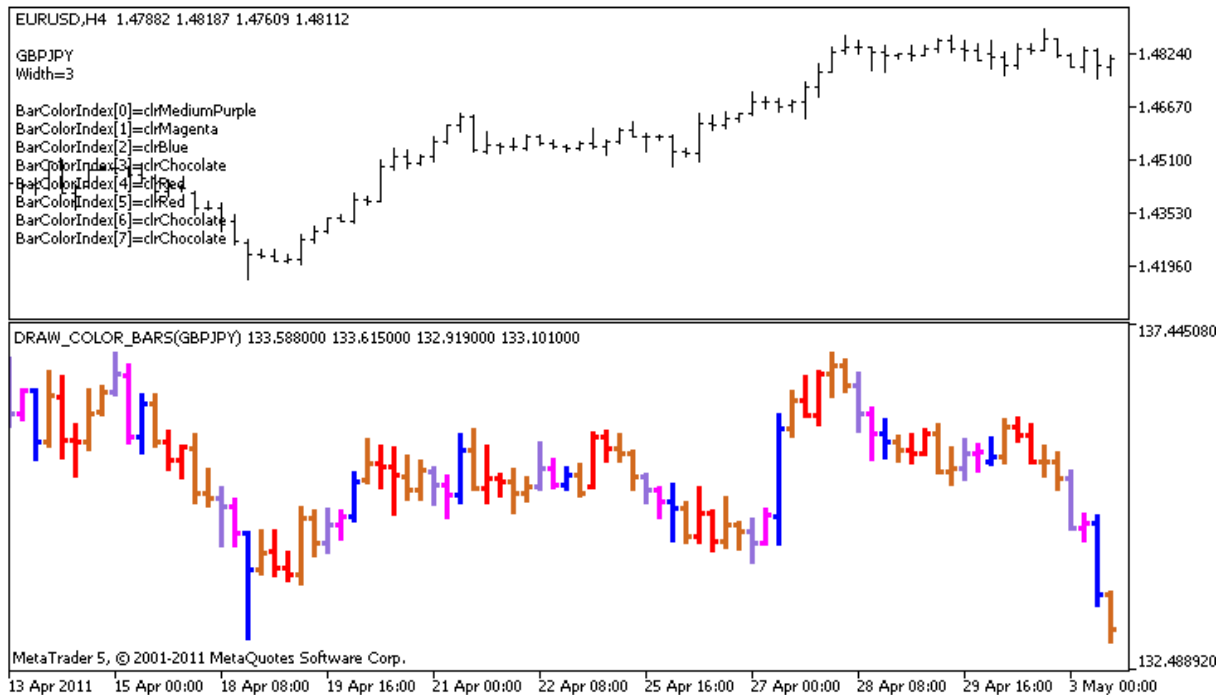
Immer füllen Sie die Indikator-Puffer mit Werten expliziert aus, geben Sie den leeren Wert in Puffer der Balken zu überspringen.

Die erforderliche Anzahl der Puffer für den Bau von DRAW\_COLOR\_BARS ist 5:

- vier Puffer zum Speichern der Werte von Open, High, Low und Close;
- einen Puffer, um die Farbindex des Balkens (es ist sinnvoll nur für gezeichnete Balken) zu speichern.

Alle Puffer sollen eine nach der anderen in dieser Reihenfolge gehen: Open, High, Low, Close und Farbpuffer. Keiner der Preis-Puffer darf nicht nur leere Werte haben, da in diesem Fall nichts gezeichnet wird.

Ein Beispiel für einen Indikator, der Balken des Finanzinstrumenten in einem separaten Fenster zeichnet. Die Balkenfarbe wird zufällig alle N Ticks verändert. Der Parameter N wird in [externen Parameter](#) des Indikators angegeben, damit kann es manuell angegeben werden (die Registerkarte "Einstellungen" im Indikatoreigenschaften-Fenster).



Bitte beachten Sie, dass für `plot1` mit dem `DRAW_COLOR_BARS`-Stil ursprünglich acht Farben mit Compiler-Direktiven `#property` angegeben werden, und dann, in der `OnCalculate()`-Funktion, wird die Farbe nach dem Zufallsprinzip aus 14 Farben, die in `Array colors[]` gespeichert sind, ausgewählt.

```
//+-----+
//|                                     DRAW_COLOR_BARS.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Der Indikator demonstriert DRAW_COLOR_BARS"
#property description "Zeichnet in einem separaten Fenster verschiedenfarbige Balken c"
#property description "Die Farbe und Größe der Balken und der Symbol werden nach dem 2"
#property description "jede N Ticks verändert"

#property indicator_separate_window
#property indicator_buffers 5
#property indicator_plots 1
//--- plot ColorBars
#property indicator_label1 "ColorBars"
#property indicator_type1  DRAW_COLOR_BARS
//--- Definieren wir 8 Farben um Balken zu färben (sie sind im speziellen Array gespei
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrLi
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- Eingabeparameter
input int      N=5;          // Anzahl der Ticks um Art zu ändern
```

```

input int      bars=500;          // Anzahl der Balken zu zeigen
input bool     messages=false;    // Ausgabe der Meldungen ins "Experts"-Journal
//--- Indicator-Puffer
double         ColorBarsBuffer1[];
double         ColorBarsBuffer2[];
double         ColorBarsBuffer3[];
double         ColorBarsBuffer4[];
double         ColorBarsColors[];
//--- Symbolname
string symbol;
int bars_colors;
//--- Array, um die Farbe zu speichern, enthält 14 Elemente
color colors[]=
{
    clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
    clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrMagenta,clrCyan,clrMediumPurple
};
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- indicator buffers mapping
    SetIndexBuffer(0,ColorBarsBuffer1,INDICATOR_DATA);
    SetIndexBuffer(1,ColorBarsBuffer2,INDICATOR_DATA);
    SetIndexBuffer(2,ColorBarsBuffer3,INDICATOR_DATA);
    SetIndexBuffer(3,ColorBarsBuffer4,INDICATOR_DATA);
    SetIndexBuffer(4,ColorBarsColors,INDICATOR_COLOR_INDEX);
    //--- Anzahl der Farben um Balken zu färben
    bars_colors=8;    // Sehen Sie. Kommentar zur Eigenschaft #property indicator_color
    //---
    return(INIT_SUCCEEDED);
};
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{
    static int ticks=0;
    //--- Zählen wir Ticks, um den Stil, die Farbe und Breite des Balkens zu ändern

```

```

    ticks++;
//--- Wenn eine ausreichende Anzahl von Ticks angesammelt hat
    if(ticks>=N)
    {
//--- Wählen wir ein neues Symbol aus dem "Market Watch"
        symbol=GetRandomSymbolName();
//--- Ändern wir die Eigenschaften der Linie
        ChangeLineAppearance();
//--- Ändern wir die Farben der Balken
        ChangeColors(colors,bars_colors);
        int tries=0;
//--- Machen wir 5 Versuche, um den Puffer mit Preise aus symbol zu füllen
        while(!CopyFromSymbolToBuffers(symbol,rates_total,bars_colors) && tries<5)
        {
            //--- Zähler der CopyFromSymbolToBuffers() Funktionsaufrufe
            tries++;
        };
//--- Setzen wir den Zähler der Ticks auf Null
        ticks=0;
    };
//--- return value of prev_calculated for next call
    return(rates_total);
};
//+-----+
//| Füllen wir den Indikator-Puffer mit Preise aus |
//+-----+
bool CopyFromSymbolToBuffers(string name,int total,int bar_colors)
{
//--- In die Array rates[] kopieren wir Preise Open, High, Low und Close
    MqlRates rates[];
//--- Zähler der Versuche
    int attempts=0;
//--- Wie viel kopiert
    int copied=0;
//--- Machen wir 25 Versuche, um die Zeitreihe für das gewünschten Symbol zu erhalten
    while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
    {
        Sleep(100);
        attempts++;
        if(messages) PrintFormat("%s CopyRates(%s) attempts=%d",__FUNCTION__,name,attempts);
    };
//--- Wenn es gelang nicht eine ausreichende Anzahl von Balken zu kopieren
    if(copied!=bars)
    {
//--- erstellen wir einen Nachrichtenstring
        string comm=StringFormat("Für Symbol %s konnte nur %d Balken von %d gefragten Balken kopiert werden",
                                   name,
                                   copied,
                                   bars);
    };
}

```



```

        );

        //--- Zeigen wir einer Nachricht in einer Kommentar auf dem Haupt-Chart-Fenster
        Comment(comm);
        //--- Anzeige von Informationen
        if(messages) Print(comm);
        return(false);
    };
else
    {
        //--- Einstellen wir Anzeige des Symbols
        PlotIndexSetString(0,PLOT_LABEL,name+" Open;" +name+" High;" +name+" Low;" +name+"
        IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_COLOR_BARS (" +name+" )");
    };
//--- Initialisieren wir den Puffer leere Werte
ArrayInitialize(ColorBarsBuffer1,0.0);
ArrayInitialize(ColorBarsBuffer2,0.0);
ArrayInitialize(ColorBarsBuffer3,0.0);
ArrayInitialize(ColorBarsBuffer4,0.0);

//--- Kopieren wir Preise in Puffer
for(int i=0;i<copied;i++)
    {
        //--- Berechnen wir den entsprechenden Index für den Puffer
        int buffer_index=total-copied+i;
        //--- Schreiben wir die Preise in Puffer
        ColorBarsBuffer1[buffer_index]=rates[i].open;
        ColorBarsBuffer2[buffer_index]=rates[i].high;
        ColorBarsBuffer3[buffer_index]=rates[i].low;
        ColorBarsBuffer4[buffer_index]=rates[i].close;
        //---
        ColorBarsColors[buffer_index]=i%bar_colors;
    };
return(true);
};
//+-----+
//| Gibt ein zufälliges Symbol aus dem Market Watch zurück |
//+-----+
string GetRandomSymbolName()
{
    //--- Anzahl der in dem "Market Watch" Fenster angezeigten Symbole
    int symbols=SymbolsTotal(true);
    //--- Position des Symbols in der Liste - eine Zufallszahl zwischen 0 und symbols
    int number=MathRand()%symbols;
    //--- Gibt den Namen des Symbols an der angegebenen Position zurück
    return SymbolName(number,true);
};
//+-----+
//| Ändert die Farbe der Segmente des ZigZags |
//+-----+

```

```

void ChangeColors(color &cols[],int plot_colors)
{
//--- Anzahl der Farben
    int size=ArraySize(cols);
//---
    string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- Für jede Farbeindex definieren wir eine neue Farbe zufällig
    for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
    {
        //--- erhalten wir eine Zufallszahl
        int number=MathRand();
        //--- Erhalten wir den Index in Array col[] als den Rest der ganzzahligen Division
        int i=number%size;
        //--- Geben wir die Farbe für jeden Index als Eigenschaft PLOT_LINE_COLOR an
        PlotIndexSetInteger(0, // Nummer der Grafikstile
                            PLOT_LINE_COLOR, // Property Identifier
                            plot_color_ind, // Farbeindex, in dem wir die Farbe setzen
                            cols[i]); // neue Farbe

        //--- schreiben wir die Farbe
        comm=comm+StringFormat("BarColorIndex[%d]=%s \r\n",plot_color_ind,ColorToString(cols[i]));
        ChartSetString(0,CHART_COMMENT,comm);
    }
//---
; }

//+-----+
//| Ändert das Aussehen von Balken |
//+-----+
void ChangeLineAppearance()
{
//--- String um Informationen über die Eigenschaften der Balken zu erstellen
    string comm="";

//--- Block mit Änderungen der Balkenbreite
    int number=MathRand();
//--- Erhalten wir die Breite als den Rest der ganzzahligen Division
    int width=number%5; // Breite ist von 0 bis 4
//--- Geben wir die Farbe als Eigenschaft PLOT_LINE_WIDTH
    PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Schreiben wir die Linienbreite
    comm=comm+"\r\nWidth="+IntegerToString(width);

//--- Schreiben wir den Namen des Symbols
    comm="\r\n"+symbol+comm;

//--- Zeigen wir Informationen auf dem Chart durch den Kommentar
    Comment(comm);
; }

```



## DRAW\_COLOR\_CANDLES

DRAW\_COLOR\_CANDLES-Stil, als [DRAW\\_CANDLES](#) zeichnet Kerzen auf den Werten der vier Indikator-Puffer, die die Preise Open, High, Low und Close enthalten. Aber außerdem, erlaubt es Ihnen, eine Farbe für jede Kerzen aus einer gegebenen Satz anzugeben. Zu diesem Zweck hat der Stil einen speziellen Farbpuffer, der Farbindizes für jeden Balken speichert. Es erlaubt individuelle Indikatoren in Form von Kerzen, darunter ein in separates Grafikfenster und andere Finanzinstrumente, zu erstellen.

Die Anzahl der Farben um die Kerzen zu färben kann durch [Compiler-Direktiven](#) oder dynamisch mit der [PlotIndexSetInteger\(\)](#)-Funktion angegeben werden. Dynamische Veränderungen in den Eigenschaften der graphischen Konstruktion ermöglichen es Ihnen, die Indikatoren zu "wiederbeleben" so, dass sie ihr Aussehen ändern je nach der aktuellen Situation.

Der Indikator ist nur für die Balken, für die nicht leer Werte der vier Indikator-Puffern angegeben wird, gezeichnet. Um anzugeben, welchen Wert als "leer" betrachtet werden soll, setzen Sie diesen Wert in der Eigenschaft [PLOT\\_EMPTY\\_VALUE](#):

```
//--- Der Leerer Wert (0) wird nicht in der Zeichnung teilnehmen  
PlotIndexSetDouble(Index_der_Darstellung_DRAW_COLOR_CANDLES, PLOT_EMPTY_VALUE, 0);
```

Immer füllen Sie die Indikator-Puffer mit Werten expliziert aus, geben Sie den leeren Wert in Puffer der Balken zu überspringen.

Die erforderliche Anzahl der Puffer für den Bau von DRAW\_COLOR\_CANDLES ist 5:

- vier Puffer zum Speichern der Werte von Open, High, Low und Close;
- einen Puffer, um die Farbindex der Kerze (es ist sinnvoll nur für gezeichnete Kerzen) zu speichern.

Alle Puffer sollen eine nach der anderen in dieser Reihenfolge gehen: Open, High, Low, Close und Farbpuffer. Keiner der Preis-Puffer darf nicht nur leere Werte haben, da in diesem Fall nichts gezeichnet wird.

Ein Beispiel für einen Indikator, der Kerzen des Finanzinstrumenten in einem separaten Fenster zeichnet. Die Kerzenfarbe wird zufällig alle N Ticks verändert. Der Parameter N wird in [externen Parameter](#) des Indikators angegeben, damit kann es manuell angegeben werden (die Registerkarte "Einstellungen" im Indikatoreigenschaften-Fenster).



Bitte beachten Sie, dass für Plot1 die Farbe mit Compiler-Direktiven [#property](#) angegeben wird, und dann in der Funktion [OnCalculate\(\)](#) wird die Farbe nach dem Zufallsprinzip aus einer vorbereiteten Liste ausgewählt.

```
//+-----+
//|                                     DRAW_COLOR_CANDLES.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "Der Indikator zeigt DRAW_COLOR_CANDLES."
#property description "Zeichnet in einem separaten Fenster verschiedenfarbige Kerzen c"
#property description " "
#property description "Die Farbe und Größe der Kerzen und der Symbol werden nach dem 2"
#property description "jede N Ticks verändert."

#property indicator_separate_window
#property indicator_buffers 5
#property indicator_plots 1
//--- plot ColorCandles
#property indicator_label1 "ColorCandles"
#property indicator_type1 DRAW_COLOR_CANDLES
//--- Definieren wir 8 Farben um Kerzen zu färben (sie sind im speziellen Array gespei
#property indicator_color1 clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrLi
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
```

```

//--- Eingabeparameter
input int      N=5;           // Anzahl der Ticks um Art zu ändern
input int      bars=500;      // Anzahl der Kerzen zu zeigen
input bool     messages=false; // Ausgabe der Meldungen ins "Experts"-Journal
//--- Indicator-Puffer
double         ColorCandlesBuffer1[];
double         ColorCandlesBuffer2[];
double         ColorCandlesBuffer3[];
double         ColorCandlesBuffer4[];
double         ColorCandlesColors[];
int            candles_colors;
//--- Symbolname
string symbol;
//--- Array, um die Farbe zu speichern, enthält 14 Elemente
color colors[] =
{
    clrRed, clrBlue, clrGreen, clrChocolate, clrMagenta, clrDodgerBlue, clrGoldenrod,
    clrIndigo, clrLightBlue, clrAliceBlue, clrMoccasin, clrMagenta, clrCyan, clrMediumPurple
};
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- wenn es gibt zu wenig bars, Arbeit früher vollenden
    if(bars<50)
    {
        Comment("Geben Sie eine größere Anzahl von Bars! Arbeit des Indikators ist beendet");
        return(INIT_PARAMETERS_INCORRECT);
    }
    //--- indicator buffers mapping
    SetIndexBuffer(0, ColorCandlesBuffer1, INDICATOR_DATA);
    SetIndexBuffer(1, ColorCandlesBuffer2, INDICATOR_DATA);
    SetIndexBuffer(2, ColorCandlesBuffer3, INDICATOR_DATA);
    SetIndexBuffer(3, ColorCandlesBuffer4, INDICATOR_DATA);
    SetIndexBuffer(4, ColorCandlesColors, INDICATOR_COLOR_INDEX);
    //--- Leerer Wert
    PlotIndexSetDouble(0, PLOT_EMPTY_VALUE, 0);
    //--- Name des Symbols, für das die Balken gezeichnet werden
    symbol = _Symbol;
    //--- Einstellen wir Anzeige des Symbols
    PlotIndexSetString(0, PLOT_LABEL, symbol+" Open;"+symbol+" High;"+symbol+" Low;"+symbol+" Close");
    IndicatorSetString(INDICATOR_SHORTNAME, "DRAW_COLOR_CANDLES("+symbol+")");
    //--- Anzahl der Farben um Kerzen zu färben
    candles_colors=8; // Sehen Sie. Kommentar zur Eigenschaft #property indicator_colors
    //---
    return(INIT_SUCCEEDED);
};
//+-----+

```

```

//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
    static int ticks=INT_MAX-100;
    //--- Zählen wir Ticks, um den Stil und die zu ändern
    ticks++;
    //--- Wenn eine ausreichende Anzahl von Ticks angesammelt hat
    if(ticks>=N)
    {
        //--- Wählen wir ein neues Symbol aus dem "Market Watch"
        symbol=GetRandomSymbolName();
        //--- Ansicht verändern
        ChangeLineAppearance();
        //--- Ändern wir die Farben der Balken
        ChangeColors(colors,candles_colors);

        int tries=0;
        //--- Machen wir 5 Versuche, um den Puffer plot1 mit Preise aus symbol zu füllen
        while(!CopyFromSymbolToBuffers(symbol,rates_total,0,
                                       ColorCandlesBuffer1,ColorCandlesBuffer2,ColorCandlesBuffer3,
                                       ColorCandlesBuffer4,ColorCandlesColors,candles_colors)
              && tries<5)
        {
            //--- Zähler der CopyFromSymbolToBuffers() Funktionsaufrufe
            tries++;
        };
        //--- Setzen wir den Zähler der Ticks auf Null
        ticks=0;
    };
    //--- return value of prev_calculated for next call
    return(rates_total);
};
//+-----+
//| Füllt die angegebene Kerze |
//+-----+
bool CopyFromSymbolToBuffers(string name,
                             int total,
                             int plot_index,
                             double &buff1[],

```

```

        double &buff2[],
        double &buff3[],
        double &buff4[],
        double &col_buffer[],
        int cndl_colors
    )

{
//--- In die Array rates[] kopieren wir Preise Open, High, Low und Close
    MqlRates rates[];
//--- Zähler der Versuche
    int attempts=0;
//--- Wieviel kopiert
    int copied=0;
//--- Machen wir 25 Versuche, um die Zeitreihe für das gewünschten Symbol zu erhalten
    while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates)<0)
    {
        Sleep(100);
        attempts++;
        if(messages) PrintFormat("%s CopyRates(%s) attempts=%d",__FUNCTION__,name,attempts);
    };
//--- Wenn es gelang nicht eine ausreichende Anzahl von Balken zu kopieren
    if(copied!=bars)
    {
        //--- erstellen wir einen Nachrichtenstring
        string comm=StringFormat("Für Symbol %s konnte nur %d Balken von %d gefragten Balken kopieren",
                                name,
                                copied,
                                bars
                                );

        //--- Zeigen wir einer Nachricht in einer Kommentar auf dem Haupt-Chart-Fenster
        Comment(comm);
        //--- Anzeige von Informationen
        if(messages) Print(comm);
        return(false);
    };
else
    {
        //--- Einstellen wir Anzeige des Symbols
        PlotIndexSetString(plot_index,PLOT_LABEL,name+" Open;" + name+" High;" + name+" Low;" + name+" Close");
        IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_COLOR_CANDLES (" + symbol + ")");
    };
//--- Initialisieren wir den Puffer leere Werte
    ArrayInitialize(buff1,0.0);
    ArrayInitialize(buff2,0.0);
    ArrayInitialize(buff3,0.0);
    ArrayInitialize(buff4,0.0);
//--- Kopieren wir Preise in den Puffer bei jedem Tick
    for(int i=0;i<copied;i++)
    {

```



```

//--- Berechnen wir den entsprechenden Index für den Puffer
    int buffer_index=total-copied+i;
//--- Schreiben wir die Preise in Puffer
    buff1[buffer_index]=rates[i].open;
    buff2[buffer_index]=rates[i].high;
    buff3[buffer_index]=rates[i].low;
    buff4[buffer_index]=rates[i].close;
//--- Geben wir die Kerzefarbe an
    int color_index=i%cndl_colors;
    col_buffer[buffer_index]=color_index;
; }
return(true);
; }
//+-----+
//| Gibt ein zufälliges Symbol aus dem Market Watch zurück |
//+-----+
string GetRandomSymbolName()
{
//--- Anzahl der in dem "Market Watch" Fenster angezeigten Symbole
    int symbols=SymbolsTotal(true);
//--- Position des Symbols in der Liste - eine Zufallszahl zwischen 0 und symbols
    int number=MathRand()%symbols;
//--- Gibt den Namen des Symbols an der angegebenen Position zurück
    return SymbolName(number,true);
; }
//+-----+
//| Ändert die Farbe der Segmente der Kerzen |
//+-----+
void ChangeColors(color &cols[],int plot_colors)
{
//--- Anzahl der Farben
    int size=ArraySize(cols);
//---
    string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- Für jede Farbeindex definieren wir eine neue Farbe zufällig
    for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
    {
        //--- erhalten wir eine Zufallszahl
        int number=MathRand();
        //--- Erhalten wir den Index in Array col[] als den Rest der ganzzahligen Divisi
        int i=number%size;
        //--- Geben wir die Farbe für jeden Index als Eigenschaft PLOT_LINE_COLOR an
        PlotIndexSetInteger(0, // Nummer der Grafikstile
            PLOT_LINE_COLOR, // Property Identifier
            plot_color_ind, // Farbeindex, in dem wir die Farbe s
            cols[i]); // neue Farbe
        //--- schreiben wir die Farbe
        comm=comm+StringFormat("CandleColorIndex[%d]=%s \r\n",plot_color_ind,ColorToStr

```

```
        ChartSetString(0, CHART_COMMENT, comm) ;  
    ; }  
//---  
    ; }  
//+-----+  
//| Ändert das Aussehen der Kerzen |  
//+-----+  
void ChangeLineAppearance()  
{  
//--- String um Informationen über die Eigenschaften der Kerzen zu erstellen  
    string comm="";  
//--- Schreiben wir den Namen des Symbols  
    comm="\r\n"+symbol+comm;  
//--- Zeigen wir Informationen auf dem Chart durch den Kommentar  
    Comment(comm) ;  
    ; }
```

## Zusammenhang zwischen Eigenschaften des Indikators und entsprechenden Funktionen

Jeder Benutzeranzeiger hat viele [Eigenschaften](#), einige von ihnen sind obligatorisch und befinden sich immer am Anfang der Beschreibung. Das sind Eigenschaften:

- Angeben des Fensters für die Darstellung des Anzeigers - `indicator_separate_window` oder `indicator_chart_window`;
- Anzahl der Indikatorpuffer - `indicator_buffers`;
- Anzahl der graphischen Darstellungen des Anzeigers - `indicator_plots`.

Aber es gibt auch andere Eigenschaften, die sowohl durch Befehlsanweisungen des [Preprozessors](#), als auch durch die Funktionen für Erzeugung des Benutzeranzeigers vorgegeben werden können. In der Tabelle werden diese Eigenschaften und ihnen entsprechende Funktionen beschrieben.

| Befehlsanweisungen für Eigenschaften des Subfensters des Anzeigers | Funktionen des Typs IndicatorSet...()                                                                                      | Beschreibung der eingestellten Eigenschaft des Subfensters            |
|--------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------|
| <code>indicator_height</code>                                      | <a href="#">IndicatorSetInteger</a><br>( <code>INDICATOR_INDICATOR_HEIGHT</code> , <code>nHeight</code> )                  | Der Wert der festen Höhe des Subfensters                              |
| <code>indicator_minimum</code>                                     | <a href="#">IndicatorSetDouble</a><br>( <code>INDICATOR_MINIMUM</code> , <code>dMaxValue</code> )                          | Minimaler Wert der vertikalen Achse                                   |
| <code>indicator_maximum</code>                                     | <a href="#">IndicatorSetDouble</a><br>( <code>INDICATOR_MAXIMUM</code> , <code>dMinValue</code> )                          | Maximaler Wert der vertikalen Achse                                   |
| <code>indicator_levelN</code>                                      | <a href="#">IndicatorSetDouble</a><br>( <code>INDICATOR_LEVELVALUE</code> , <code>N-1</code> , <code>nLevelValue</code> )  | Wert der vertikalen Achse für N Level                                 |
| keine Befehlsanweisung des Preprozessors                           | <a href="#">IndicatorSetString</a><br>( <code>INDICATOR_LEVELTEXT</code> , <code>N-1</code> , <code>sLevelName</code> )    | Name des dargestellten Levels                                         |
| <code>indicator_levelcolor</code>                                  | <a href="#">IndicatorSetInteger</a><br>( <code>INDICATOR_LEVELCOLOR</code> , <code>N-1</code> , <code>nLevelColor</code> ) | Farbe für die Darstellung des N Level                                 |
| <code>indicator_levelwidth</code>                                  | <a href="#">IndicatorSetInteger</a><br>( <code>INDICATOR_LEVELWIDTH</code> , <code>N-1</code> , <code>nLevelWidth</code> ) | Breite der Linie für die Darstellung des n Levels                     |
| <code>indicator_levelstyle</code>                                  | <a href="#">IndicatorSetInteger</a><br>( <code>INDICATOR_LEVELSTYLE</code> , <code>N-1</code> , <code>nLevelStyle</code> ) | Stil der Linie für die Darstellung des N Levels                       |
| Befehlsanweisungen für graphische Darstellungen                    | Funktionen des Typs PlotIndexSet...()                                                                                      | Beschreibung der eingestellten Eigenschaft für graphische Darstellung |

|                                               |                                                                                         |                                                                                                                                                                                                                                                                                                                              |
|-----------------------------------------------|-----------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| indicator_labelN                              | <a href="#">PlotIndexSetString</a> (N-1, <a href="#">PLOT_LABEL</a> ,sLabel)            | Kurze Benennung für die graphische Darstellung mit Nummer N. Gezeigt im Fenster DataWindow und im pop-up tooltip beim bei Cursor-Bewegung darauf                                                                                                                                                                             |
| indicator_colorN                              | <a href="#">PlotIndexSetInteger</a> (N-1, <a href="#">PLOT_LINE_COLOR</a> , nColor)     | Liniefarbe für graphische Darstellung mit Nummer N                                                                                                                                                                                                                                                                           |
| indicator_styleN                              | <a href="#">PlotIndexSetInteger</a> (N-1, <a href="#">PLOT_LINE_STYLE</a> , nType)      | Liniestil für für graphische Darstellung mit Nummer N                                                                                                                                                                                                                                                                        |
| indicator_typeN                               | <a href="#">PlotIndexSetInteger</a> (N-1, <a href="#">PLOT_DRAW_TYPE</a> , nType)       | Linietyt für graphische Darstellung mit Nummer N                                                                                                                                                                                                                                                                             |
| indicator_widthN                              | <a href="#">PlotIndexSetInteger</a> (N-1, <a href="#">PLOT_LINE_WIDTH</a> , nWidth)     | Breite der Linie für graphische Darstellung mit Nummer N                                                                                                                                                                                                                                                                     |
| <b>Allgemeine Eigenschaften des Anzeigers</b> | <b>Funktionen des Typs IndicatorSet...()</b>                                            | <b>Beschreibung</b>                                                                                                                                                                                                                                                                                                          |
| keine Befehlsanweisung des Preprozessors      | <a href="#">IndicatorSetString</a> ( <a href="#">INDICATOR_SHORTNAME</a> , sShortName)  | Stellt passenden kurzen Namen des Anzeigers, der in der Liste der Anzeiger dargestellt werden wird (im Terminal wird durch <b>Ctrl+I</b> aufgerufen)).                                                                                                                                                                       |
| keine Befehlsanweisung des Preprozessors      | <a href="#">IndicatorSetInteger</a> ( <a href="#">INDICATOR_DIGITS</a> , nDigits)       | Stellt die angeforderte Genauigkeit für Darstellung der Anzeigerwerte - Anzahl der Dezimalzeichen                                                                                                                                                                                                                            |
| keine Befehlsanweisung des Preprozessors      | <a href="#">IndicatorSetInteger</a> ( <a href="#">INDICATOR_LEVELS</a> , nLevels)       | Stellt die Anzahl der Levels im Fenster des Anzeigers ein                                                                                                                                                                                                                                                                    |
| indicator_applied_price                       | Keine Funktion, Eigenschaft wird nur durch die Anweisung des Preprozessors eingestellt) | Default-Preistyp, der für Berechnung der Anzeigerwerte verwendet wird. Angegeben nur bei der Notwendigkeit, die Funktion <a href="#">OnCalculate()</a> des ersten Typs zu verwenden. Wert der Eigenschaft kann auch vom Dialog der Anzeigereigenschaften in "Parameter" - " <a href="#">Anwenden zu</a> " vorgegeben werden. |

Es muss bemerkt werden, dass Numerierung der Levels und graphischer Darstellungen in Termini des Preprozessors mit eins anfaengt, während die Numerierung derselben Eigenschaften bei der Verwendung der Funktionen mit Null anfaengt, d.h. der angegebene Wert muss um 1 kleiner sein als für #property.

Es gibt einige Befehlsanweisungen, für denen keine entsprechenden Funktionen existieren:

| Befehlsanweisung          | Beschreibung                                                        |
|---------------------------|---------------------------------------------------------------------|
| indicator_chart_window    | Anzeiger wird im Hauptfenster dargestellt                           |
| indicator_separate_window | Anzeiger wird im selbststaendigen Subfenster dargestellt            |
| indicator_buffers         | Gibt die Anzahl der erforderlichen Anzeigerbuffer an                |
| indicator_plots           | Gibt die Anzahl der <u>graphischen Darstellungen</u> im Anzeiger an |

## SetIndexBuffer

Verbindet den angegebenen Indikatorpuffer mit einem eindimensionalen dynamischen Feld des Typs double.

```
bool SetIndexBuffer(
    int          index,          // Index des Puffers
    double       buffer[],       // Feld
    ENUM_INDEXBUFFER_TYPE data_type // was wird aufbewahren
);
```

### Parameter

*index*

[in] Nummer des Indikatorpuffers. Numerierung faengt mit 0 an. Nummer muss kleiner sein als der Wert, der in [#property indicator\\_buffers](#) erklärt wurde.

*buffer[]*

[in] Feld, das im Programm des Benutzerindikators erklärt wird.

*data\_type*

[in] Typ der Daten, die im Indikatorfeld aufbewahren werden. Default-Wert ist [INDICATOR\\_DATA](#) (Werte des berechneten Indikators). Kann auch den Wert [INDICATOR\\_COLOR\\_INDEX](#) annehmen, dann wird der vorliegende Puffer für Aufbewahrung der Farbenindexe für den vorherigen Indikatorpuffer. Man kann bis 64 [Farben](#) in der Zeile [#property indicator\\_colorN](#) vorgeben. Wert [INDICATOR\\_CALCULATIONS](#) bedeutet dass der vorliegende Puffer für Zwischenberechnungen des Indikators verwendet wird und für Zeichnen nicht bestimmt ist.

### Rückgabewert

Im Erfolgsfall gibt [true](#) zurück, anderenfalls [false](#).

### Bemerkung

Nach der Bindung wird das dynamische Feld *buffer[]* dieselbe Indizierung haben, wie in normalen Feldern, auch wenn für das gebundene Feld Indizierung wie in [Timeserien](#) vorher eingestellt wird. Wenn es notwendig ist, die Folge des Zugangs zu Elementen des Indikatorfeldes zu verändern, muss man die Funktion [ArraySetAsSeries\(\)](#) nach der Bindung des Feldes durch die Funktion [SetIndexBuffer\(\)](#) anwenden. Dabei muss man behalten, dass die Größe der dynamischen Felder, die als Indikatorpuffer durch die Funktion [SetIndexBuffer\(\)](#) zugeordnet werden, nicht verändert werden muss. Für Indikatorpuffer werden alle Operationen der Größenveränderung vom ausführenden Untersystem des Terminals durchgeführt.

### Beispiel:

```
//+-----+
//|                                     TestCopyBuffer1.mq5 |
//|                                     Copyright 2009, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+

#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

```

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots 1
//---- plot MA
#property indicator_label1 "MA"
#property indicator_type1 DRAW_LINE
#property indicator_color1 clrRed
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//--- input parameters
input bool AsSeries=true;
input int period=15;
input ENUM_MA_METHOD smootMode=MODE_EMA;
input ENUM_APPLIED_PRICE price=PRICE_CLOSE;
input int shift=0;
//--- indicator buffers
double MABuffer[];
int ma_handle;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- indicator buffers mapping
if(AsSeries) ArraySetAsSeries(MABuffer,true);
Print("Anzeigerpuffer ist eine Zeitreihe = ",ArrayGetAsSeries(MABuffer));
SetIndexBuffer(0,MABuffer,INDICATOR_DATA);
Print("Anzeigerpuffer nach SetIndexBuffer() ist eine Zeitreihe = ",
ArrayGetAsSeries(MABuffer));

//--- verändern wir die Folge des Zugangs zu Elementen des Anzeigerpuffers
ArraySetAsSeries(MABuffer,AsSeries);

IndicatorSetString(INDICATOR_SHORTNAME,"MA("+period+")"+AsSeries);
//---
ma_handle=ima(Symbol(),0,period,shift,smootMode,price);
return(INIT_SUCCEEDED);
;}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],

```

```
        const long &tick_volume[],
        const long &volume[],
        const int &spread[])
{
//--- kopieren wir die Werte des gleitenden Mittelwertes in Puffer MABuffer
    int copied=CopyBuffer(ma_handle,0,0,rates_total,MABuffer);

    Print("MABuffer[0] = ",MABuffer[0]); // abhängig vom Wert AsSeries
                                         // bekommen wir entweder den aeltesten Wert
                                         // oder für die laufende unbeendete Bar

//--- return value of prev_calculated for next call
    return(rates_total);
;}
//+-----+
```

#### Sehen Sie auch

[Eigenschaften der Benutzerindikatoren](#) , [Zugang zu Zeitreihen und Indikatoren](#)



## IndicatorSetDouble

Gibt den Wert der entsprechenden Indikatoreigenschaft. Indikatoreigenschaft muss des Typs double sein. Es gibt 2 Varianten der Funktion.

### Aufruf mit dem Identifikator der Eigenschaft

```
bool IndicatorSetDouble (
    int      prop_id,           // Identifikator
    double   prop_value        // der eingestellte Wert
);
```

### Aufruf mit dem Identifikator und Modifikator der Eigenschaft

```
bool IndicatorSetDouble (
    int      prop_id,           // Identifikator
    int      prop_modifier,    // Modifikator
    double   prop_value        // der eingestellte Wert
);
```

### Parameter

*prop\_id*

[in] Identifikator der Indikatoreigenschaft. Wert kann einer der Aufzählungswerte [ENUM\\_CUSTOMIND\\_PROPERTY\\_DOUBLE](#) sein.

*prop\_modifier*

[in] Modifikator der angegebenen Eigenschaft. Nur Leveleigenschaften fordern Modifikator an. Nummerierung der Stände beginnt bei 0, d.h. um eine Eigenschaft dem zweiten Stand anzugeben, setzen sie 1 (1 weniger als bei der Verwendung von [Compiler-Direktiven](#)).

*prop\_value*

[in] Wert der Eigenschaft.

### Rückgabewert

Bei der erfolgreichen Durchführung gibt true zurück, anderenfalls false.

### Hinweis

Nummerierung der Eigenschaften (Modifikatoren) durch Direktive #property beginnt bei 1 (eins), während die Funktionen verwenden Nummerierung bei 0 (Null). Bei die falsche Eingabe einer Nummer des Stands, kann [Anzeige des Indikators](#) verschieden sein.

Können Sie beispielsweise den Wert des ersten Stands für den Indikator in einem separaten Fenster auf zwei Methoden angeben:

- property indicator\_level1 50 - 1 wird für Angabe des Stands verwendet,
- IndicatorSetDouble(INDICATOR\_LEVELVALUE, 0, 50) - 0 wird für Angabe des Stands verwendet.

**Beispiel:** umwendender Indikator, der den maximalen und minimalen Wert des Indikatorfensters und die Werte der Stände, an denen die horizontalen Linien liegen, ändert.



```

//--- die maximalen und minimalen Werte für den Indikatorfenster setzen
#property indicator_minimum 0
#property indicator_maximum 100
//--- Anzeige der drei horizontalen Stände in einem separaten Indikatorfenster angeben
#property indicator_level1 25
#property indicator_level2 50
#property indicator_level3 75
//--- Breite der Stände angeben
#property indicator_levelwidth 1
//--- Stil der horizontalen Stände angeben
#property indicator_levelstyle STYLE_DOT
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- Beschreibung der horizontalen Stände angeben
IndicatorSetString(INDICATOR_LEVELTEXT,0,"First Level (index 0)");
IndicatorSetString(INDICATOR_LEVELTEXT,1,"Second Level (index 1)");
IndicatorSetString(INDICATOR_LEVELTEXT,2,"Third Level (index 2)");
//--- Kurzname des Indikators angeben
IndicatorSetString(INDICATOR_SHORTNAME,"IndicatorSetDouble() Demo");
//--- Individuelle Farbe für die St'nde angeben
IndicatorSetInteger(INDICATOR_LEVELCOLOR,0,clrBlue);
IndicatorSetInteger(INDICATOR_LEVELCOLOR,1,clrGreen);
IndicatorSetInteger(INDICATOR_LEVELCOLOR,2,clrRed);
//---
return(INIT_SUCCEEDED);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
static int tick_counter=0;
static double level1=25,level2=50,level3=75;
static double max=100,min=0, shift=100;
//--- Ticks berechnen
tick_counter++;
//--- Umwälzen auf jedem zehnten Tick
if(tick_counter%10==0)
{
//--- umwenden Zeichen für Standwerte
level1=-level1;
level2=-level2;
level3=-level3;
//--- umwenden Zeichen für maximale und minimale Werte
max-=shift;
min-=shift;
//--- umwenden Wert von Verschiebung
shift=-shift;
//--- neue Werte der Stände angeben

```

```
IndicatorSetDouble(INDICATOR_LEVELVALUE,0,level1);
IndicatorSetDouble(INDICATOR_LEVELVALUE,1,level2);
IndicatorSetDouble(INDICATOR_LEVELVALUE,2,level3);
//--- Neue maximale und minimale Werte für Indikatorfenster angeben
Print("Set up max = ",max," min = ",min);
IndicatorSetDouble(INDICATOR_MAXIMUM,max);
IndicatorSetDouble(INDICATOR_MINIMUM,min);
}
//--- return value of prev_calculated for next call
return(rates_total);
}
```

#### Sehen Sie auch

[Stile der Indikator in den Beispielen](#), [Zusammenhang zwischen Eigenschaften des Indikators und entsprechenden Funktionen](#), [Zeichnungsstile](#)

## IndicatorSetInteger

Gibt den Wert der entsprechenden Eigenschaft des Indikators vor. Eigenschaft des Indikators muss des Typ int oder color sein. Es gibt 2 Varianten der Funktion.

**Aufruf mit Andeutung der Eigenschaft des Identifikatoren.**

```
bool IndicatorSetInteger (
    int prop_id,           // Identifikator
    int prop_value         // der eingestellte Wert
);
```

**Aufruf mit Andeutung des Identifikatoren und Modifikatoren der Eigenschaft**

```
bool IndicatorSetInteger (
    int prop_id,           // Identifikator
    int prop_modifier,     // Modifikator
    int prop_value         // der eingestellte Wert
);
```

### Parameter

*prop\_id*

[in] Identifikator der Indikatoreigenschaft. Wert kann einer der Aufzählungswerte [ENUM\\_CUSTOMIND\\_PROPERTY\\_INTEGER](#) sein.

*prop\_modifier*

[in] Modifikator der angegebenen Eigenschaft. Nur Leveleigenschaften erfordern einen Modifikator.

*prop\_value*

[in] Wert der Eigenschaft.

### Rückgabewert

Im Erfolgsfall gibt true zurück, anderenfalls false.

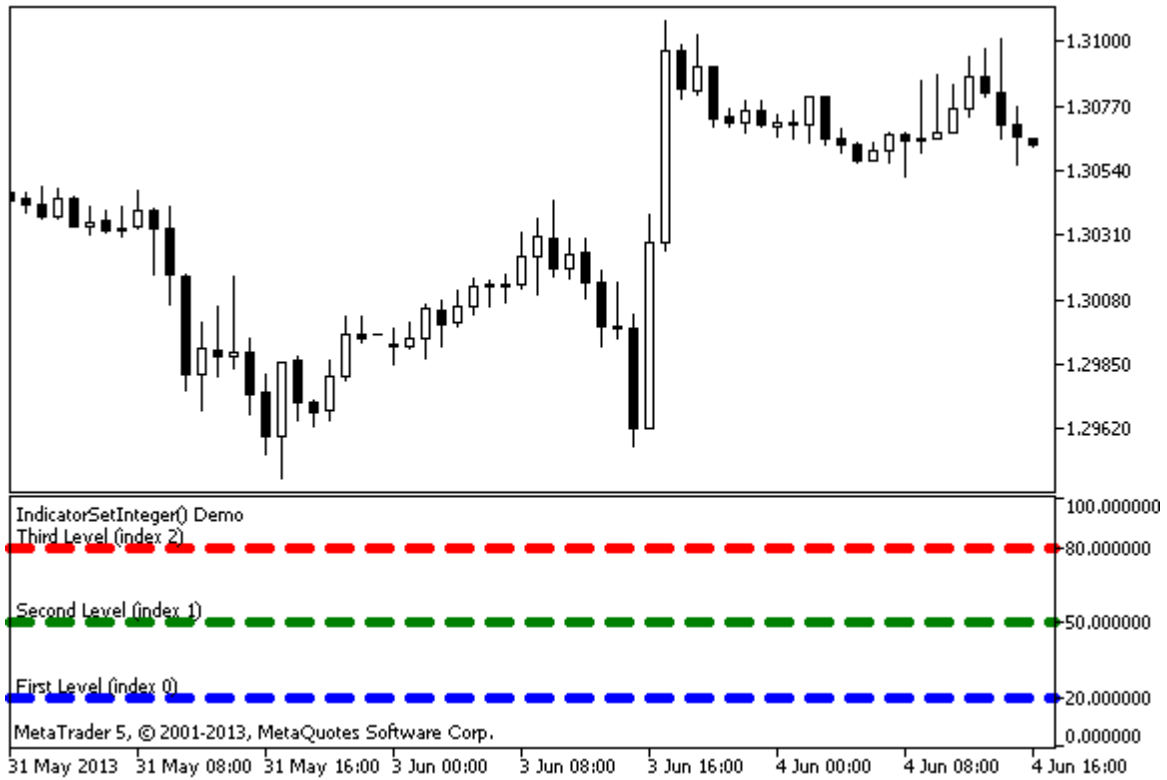
### Hinweis

Nummerierung der Eigenschaften (Modifikatoren) durch Direktive #property beginnt bei 1 (eins), während die Funktion verwendet Nummerierung bei 0 (Null). Bei die falsche Eingabe einer Nummer des Stands, kann [Anzeige des Indikatoren](#) verschieden sein.

Zum Beispiel, um die Dicke der ersten horizontalen Ebene anzugeben, verwenden Sie Null-Index:

- IndicatorSetInteger(INDICATOR\_LEVELWIDTH, 0, 5) - 0 wird für Angabe des Linienbreite für ersten Stand verwendet.

**Beispiel:** der Indikator setzt die Farbe, den Stil und die Breite der horizontalen Linien des Indikators.



```

#property indicator_separate_window
#property indicator_minimum 0
#property indicator_maximum 100
//--- Anzeige der horizontalen Stände des Indikators in einem separaten Fenster
#property indicator_level1 20
#property indicator_level2 50
#property indicator_level3 80
//--- Breite der horizontalen Linien angeben
#property indicator_levelwidth 5
//--- Farbe der horizontalen Linien angeben=
#property indicator_levelcolor clrAliceBlue
//--- Stil der horizontalen Linien angeben
#property indicator_levelstyle STYLE_DOT
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- Beschreibung der horizontalen Linien angeben
IndicatorSetString(INDICATOR_LEVELTEXT,0,"First Level (index 0)");
IndicatorSetString(INDICATOR_LEVELTEXT,1,"Second Level (index 1)");
IndicatorSetString(INDICATOR_LEVELTEXT,2,"Third Level (index 2)");
//--- Kurzname für Indikator angeben
IndicatorSetString(INDICATOR_SHORTNAME,"IndicatorSetInteger() Demo");
return(INIT_SUCCEEDED);
}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
    static int tick_counter=0;
//--- Ticks berechnen
    tick_counter++;
//--- Farben der horizontalen Linien abhängig vom Tick-Zähler
    ChangeLevelColor(0,tick_counter,3,6,10); // drei letzte Parameter ändern die Farbe
    ChangeLevelColor(1,tick_counter,3,6,8);
    ChangeLevelColor(2,tick_counter,4,7,9);
//--- Stil der horizontalen Linien ändern
    ChangeLevelStyle(0,tick_counter);
    ChangeLevelStyle(1,tick_counter+5);
    ChangeLevelStyle(2,tick_counter+15);
//--- Linienbreite wie der Rest der ganzzahligen Division der Anzahl von Ticks bei 5
    int width=tick_counter%5;
//--- Durch alle horizontalen Ebenen gehen und angeben
    for(int l=0;l<3;l++)
        IndicatorSetInteger(INDICATOR_LEVELWIDTH,l,width+1);
//--- return value of prev_calculated for next call
    return(rates_total);
}
//+-----+
//| Wir geben die Farbe der horizontale Linie im separaten Indikatorfenster an |

```

```
//+-----+
void ChangeLevelColor(int level,          // Nummer der horizontale Linie
                     int tick_number, // die Dividende, um einen Rest der Division zu
                     int f_trigger,  // erster Teiler von Farbschalter
                     int s_trigger,  // zweiter Teiler von Farbschalter
                     int t_trigger) // dritter Teiler von Farbschalter
{
    static color colors[3]={clrRed,clrBlue,clrGreen};
    //--- Farbindex aus Array colors[]
    int index=-1;
    //--- Berechnung der Farbnummer in Array colors[] um die horizontale Linie zu malen
    if(tick_number%f_trigger==0)
        index=0; // wenn die Zahl tick_number ist teilbar durch f_trigger
    if(tick_number%s_trigger==0)
        index=1; // wenn die Zahl tick_number ist teilbar durch s_trigger
    if(tick_number%t_trigger==0)
        index=2; // wenn die Zahl tick_number ist teilbar durch t_trigger
    //--- wenn die Farbe definiert ist, setzen wir sie
    if(index!=-1)
        IndicatorSetInteger(INDICATOR_LEVELCOLOR,level,colors[index]);
    //---
}
//+-----+
//| Stil der horizontale Linie im separaten Indikatorfenster angeben |
//+-----+
void ChangeLevelStyle(int level,          // Nummer der horizontale Linie
                     int tick_number // Zahl um Rest der Division zu erhalten)
{
    //--- Array um Stile zu speichern
    static ENUM_LINE_STYLE styles[5]=
        {STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOTDOT};
    //--- Stilindex aus Array styles[]
    int index=-1;
    //--- Berechnung der Nummer aus Array styles[] um den Stil der horizontalen Linie zu s
    if(tick_number%50==0)
        index=5; // wenn die Zahl tick_number ist durch 50 teilbar, dann Stil STYLE_D
    if(tick_number%40==0)
        index=4; // ... Stil STYLE_DASHDOT
    if(tick_number%30==0)
        index=3; // ... STYLE_DOT
    if(tick_number%20==0)
        index=2; // ... STYLE_DASH
    if(tick_number%10==0)
        index=1; // ... STYLE_SOLID
    //--- wenn Stil definiert ist, setzen wir ihn
    if(index!=-1)
        IndicatorSetInteger(INDICATOR_LEVELSTYLE,level,styles[index]);
}
}
```

Sehen Sie auch

[Eigenschaften der Benutzerobjekte](#), [Programmeigenschaften \(#property\)](#), [Zeichnungsstile](#)



## IndicatorSetString

Gibt den Wert der entsprechenden Eigenschaft des Indikators vor. Eigenschaft des Indikators muss des Typs string sein. Es gibt 2 Varianten der Funktion.

**Aufruf mit Andeutung des Identifikatoren der Eigenschaft.**

```
bool IndicatorSetString(  
    int      prop_id,           // Identifikator  
    string   prop_value        // der eingestellte Wert  
);
```

**Aufruf mit Andeutung des Identifikatoren und Modifikatoren der Eigenschaft.**

```
bool IndicatorSetString(  
    int      prop_id,           // Identifikator  
    int      prop_modifier,     // Modifikator  
    string   prop_value        // der eingestellte Wert  
);
```

### Parameter

*prop\_id*

[in] Identifikator der Eigenschaft des Indikators. Wert kann einer der Aufzählungswerte [ENUM\\_CUSTOMIND\\_PROPERTY\\_STRING](#) sein.

*prop\_modifier*

[in] Modifikator der angegebenen Eigenschaft. Nur Leveleigenschaften erfordern einen Modifikator.

*prop\_value*

[in] Wert der Eigenschaft.

### Rückgabewert

Im Erfolgsfall gibt true zurück, anderenfalls false.

### Hinweis

Nummerierung der Eigenschaften (Modifikatoren) durch Direktive #property beginnt bei 1 (eins), während die Funktion verwendet Nummerierung bei 0 (Null). Bei die falsche Eingabe einer Nummer des Stands, kann [Anzeige des Indikatoren](#) verschieden sein.

Zum Beispiel, um die Beschreibung der ersten horizontalen Ebene anzugeben, verwenden Sie Null-Index:

- IndicatorSetString(INDICATOR\_LEVELTEXT, 0, "First Level") - 0 wird für Angabe des Textbeschreibung für ersten Stand verwendet.

**Beispiel:** der Indikator setzt Texte für horizontale Linien des Indikatoren.



```
#property indicator_separate_window
#property indicator_minimum 0
#property indicator_maximum 100
//--- Anzeige der Indikatorebene in einem separaten Indikatorfenster angeben
#property indicator_level1 30
#property indicator_level2 50
#property indicator_level3 70
//--- Farbe der horizontale Linien angeben
#property indicator_levelcolor clrRed
//--- Stil der horizontale Linien angeben
#property indicator_levelstyle STYLE_SOLID
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- Beschreibungen der horizontale Linien angeben
    IndicatorSetString(INDICATOR_LEVELTEXT,0,"First Level (index 0)");
    IndicatorSetString(INDICATOR_LEVELTEXT,1,"Second Level (index 1)");
    IndicatorSetString(INDICATOR_LEVELTEXT,2,"Third Level (index 2)");
//--- Kurzname des Indikators angeben
    IndicatorSetString(INDICATOR_SHORTNAME,"IndicatorSetString() Demo");
//---
    return(INIT_SUCCEEDED);
};
//+-----+
//| Custom indicator iteration function |
```

```
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
//---

//--- return value of prev_calculated for next call
    return(rates_total);
};
```

Sehen Sie auch

[Eigenschaften der Benutzerobjekte](#), [Programmeigenschaften \(#property\)](#)

## PlotIndexSetDouble

Gibt den Wert der entsprechenden Eigenschaft der entsprechenden Indikatorlinie vor. Eigenschaft des Indikators muss des Typs double sein.

```
bool PlotIndexSetDouble(  
    int    plot_index,    // Index des graphischen Stils  
    int    prop_id,       // Identifikator der Eigenschaft  
    double prop_value     // der eingestellte Wert  
);
```

### Parameter

*plot\_index*

[in] Index der [graphischen Darstellung](#)

*prop\_id*

[in] Identifikator der Indikatoreigenschaft. Wert kann einer der Aufzählungswerte [ENUM\\_PLOT\\_PROPERTY\\_DOUBLE](#) sein.

*prop\_value*

[in] Wert der Eigenschaft.

### Rückgabewert

Bei der erfolgreichen Durchführung gibt true zurück, anderenfalls false.

## PlotIndexSetInteger

Gibt den Wert der entsprechenden Eigenschaft des Indikators vor. Eigenschaft des Indikators muss des Typs int, char, bool oder color sein. Es gibt 2 Varianten der Funktion.

**Aufruf mit Andeutung des Identifikatoren der Eigenschaft.**

```
bool PlotIndexSetInteger (
    int  plot_index,      // Index des graphischen Stils
    int  prop_id,         // Identifikator der Eigenschaft
    int  prop_value       // der eingestellte Wert
);
```

**Aufruf mit Andeutung des Identifikatoren und Modifikatoren der Eigenschaft.**

```
bool PlotIndexSetInteger (
    int  plot_index,      // Index des graphischen Stils
    int  prop_id,         // Identifikator der Eigenschaft
    int  prop_modifier,   // Modifikator der Eigenschaft
    int  prop_value       // der eingestellte Wert
);
```

### Parameter

*plot\_index*

[in] Index der [graphischen Darstellung](#)

*prop\_id*

[in] Identifikator der Indikatoreigenschaft. Wert kann einer der Aufzählungswerte [ENUM\\_PLOT\\_PROPERTY\\_INTEGER](#) sein.

*prop\_modifier*

[in] Modifikator der angegebenen Eigenschaft. Nur Eigenschaften der Indexe erfordern einen Modifikator.

*prop\_value*

[in] Wert der Eigenschaft.

### Rueckgabewert

Im Erfolgsfall gibt true zurück, anderenfalls false.

**Beispiel:** Indikator, der eine dreifarbige Linie zeichnet. Farbschema verändert sich wird jede 5 Ticks.



```
#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots 1
//---- plot ColorLine
#property indicator_label1 "ColorLine"
#property indicator_type1 DRAW_COLOR_LINE
#property indicator_color1 clrRed,clrGreen,clrBlue
#property indicator_style1 STYLE_SOLID
#property indicator_width1 3
//--- indicator buffers
double ColorLineBuffer[];
double ColorBuffer[];
int MA_handle;
//+-----+
//| Custom indicator initialization function |
//+-----+
void OnInit()
{
//--- indicator buffers mapping
SetIndexBuffer(0,ColorLineBuffer,INDICATOR_DATA);
SetIndexBuffer(1,ColorBuffer,INDICATOR_COLOR_INDEX);
//--- get MA handle
MA_handle=iMA(Symbol(),0,10,0,MODE_EMA,PRICE_CLOSE);
//---
;}
//+-----+
//| get color index |
//+-----+
```

```

int getIndexOfColor(int i)
{
    int j=i%300;
    if(j<100) return(0); // first index
    if(j<200) return(1); // second index
    return(2); // third index
};
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
    //---
    static int ticks=0,modified=0;
    int limit;
    //--- first calculation or number of bars was changed
    if(prev_calculated==0)
    {
        //--- copy values of MA into indicator buffer ColorLineBuffer
        int copied=CopyBuffer(MA_handle,0,0,rates_total,ColorLineBuffer);
        if(copied<=0) return(0); // copying failed - throw away
        //--- now set line color for every bar
        for(int i=0;i<rates_total;i++)
            ColorBuffer[i]=getIndexOfColor(i);
    };
    else
    {
        //--- copy values of MA into indicator buffer ColorLineBuffer
        int copied=CopyBuffer(MA_handle,0,0,rates_total,ColorLineBuffer);
        if(copied<=0) return(0);

        ticks++; // ticks counting
        if(ticks>=5) // it's time to change color scheme
        {
            ticks=0; // reset counter
            modified++; // counter of color changes
            if(modified>=3) modified=0; // reset counter
            ResetLastError();
            switch(modified)
            {

```

```

        case 0:// first color scheme
            PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrRed);
            PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrBlue);
            PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrGreen);
            Print("Color scheme "+modified);
            break;
        case 1:// second color scheme
            PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrYellow);
            PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrPink);
            PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrLightSlateGray);
            Print("Color scheme "+modified);
            break;
        default:// third color scheme
            PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrLightGoldenrod);
            PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrOrchid);
            PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrLimeGreen);
            Print("Color scheme "+modified);
    };
};
else
{
    //--- set start position
    limit=prev_calculated-1;
    //--- now we set line color for every bar
    for(int i=limit;i<rates_total;i++)
        ColorBuffer[i]=getIndexOfColor(i);
};
};
//--- return value of prev_calculated for next call
return(rates_total);
};
//+-----+

```



## PlotIndexSetString

Gibt den Wert der entsprechenden Eigenschaft der entsprechenden Indikatorlinie zurück. Indikatoreigenschaft muss des Typs string sein.

```
bool PlotIndexSetString(  
    int     plot_index,    // Index des graphischen Stils  
    int     prop_id,       // Identifikator der Eigenschaft  
    string  prop_value     // der eingestellte Wert  
);
```

### Parameter

*plot\_index*

[in] Index der [graphischen Darstellung](#)

*prop\_id*

[in] Identifikator der Indikatoreigenschaft. Wert kann einer der Aufzählungswerte [ENUM\\_PLOT\\_PROPERTY\\_STRING](#) sein.

*prop\_value*

[in] Eigenschaftswert.

### Rückgabewert

Im Erfolgsfall gibt true zurück, anderenfalls false.

## PlotIndexGetInteger

Gibt den Wert der entsprechenden Eigenschaft der entsprechenden Indikatorlinie zurück. Eigenschaft muss der Typen `int`, `color`, `bool` oder `char` sein. Es gibt 2 Varianten der Funktion.

**Aufruf mit Andeutung des Identifikatoren der Eigenschaft.**

```
int PlotIndexGetInteger (
    int plot_index,          // Index des graphischen Stils
    int prop_id,             // Identifikator der Eigenschaft
);
```

**Aufruf mit Andeutung des Identifikatoren und Modifikatoren der Eigenschaft .**

```
int PlotIndexGetInteger (
    int plot_index,          // Index des graphischen Stils
    int prop_id,             // Identifikator der Eigenschaft
    int prop_modifier        // Modifikator der Eigenschaft
)
```

### Parameter

*plot\_index*

[in] Index der [graphischen Darstellung](#)

*prop\_id*

[in] Identifikator der Indikatoreigenschaft. Wert kann einer der Aufzählungswerte [ENUM\\_PLOT\\_PROPERTY\\_INTEGER](#) sein.

*prop\_modifier*

[in] Modifikator der angegebenen Eigenschaft. Nur Eigenschaften der Farbeneindeix erfordert einen Modifikator.

### Bemerkung

Funktion ist für Extrahieren der Zeicheneinstellungen der entsprechenden Indikatorlinie bestimmt. Funktion arbeitet mit der Funktion [PlotIndexSetInteger](#) für Kopieren der Indikatoreigenschaften von einer Linie in die andere.

**Beispiel:** Indikator, der Kerzen in Farben zeichnet abhängig von dem Wochentag. Farben für jeden Tag werden programmweise vorgegeben.



```
#property indicator_separate_window
#property indicator_buffers 5
#property indicator_plots 1
//---- plot ColorCandles
#property indicator_label1 "ColorCandles"
#property indicator_type1 DRAW_COLOR_CANDLES
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//--- indicator buffers
double      OpenBuffer[];
double      HighBuffer[];
double      LowBuffer[];
double      CloseBuffer[];
double      ColorCandlesColors[];
color       ColorOfDay[6]={CLR_NONE,clrMediumSlateBlue,
                           clrDarkGoldenrod,clrForestGreen,clrBlueViolet,clrRed};

//+-----+
//| Custom indicator initialization function |
//+-----+
void OnInit()
{
//--- indicator buffers mapping
SetIndexBuffer(0,OpenBuffer,INDICATOR_DATA);
SetIndexBuffer(1,HighBuffer,INDICATOR_DATA);
SetIndexBuffer(2,LowBuffer,INDICATOR_DATA);
SetIndexBuffer(3,CloseBuffer,INDICATOR_DATA);
SetIndexBuffer(4,ColorCandlesColors,INDICATOR_COLOR_INDEX);
//--- set number of colors in color buffer
```

```

    PlotIndexSetInteger(0,PLOT_COLOR_INDEXES,6);
//--- set colors for color buffer
    for(int i=1;i<6;i++)
        PlotIndexSetInteger(0,PLOT_LINE_COLOR,i,ColorOfDay[i]);
//--- set accuracy
    IndicatorSetInteger(INDICATOR_DIGITS,_Digits);
    printf("We have %u colors of days",PlotIndexGetInteger(0,PLOT_COLOR_INDEXES));
//---
; }
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
//---
    int i;
    MqlDateTime t;
//----
    if(prev_calculated==0) i=0;
    else i=prev_calculated-1;
//----
    while(i<rates_total)
    {
        OpenBuffer[i]=open[i];
        HighBuffer[i]=high[i];
        LowBuffer[i]=low[i];
        CloseBuffer[i]=close[i];
        //--- set color for every candle
        TimeToStruct(time[i],t);
        ColorCandlesColors[i]=t.day_of_week;
        //---
        i++;
    }
//--- return value of prev_calculated for next call
    return(rates_total);
; }
//+-----+

```

## Graphische Objekte

Gruppe der Funktionen, die für Arbeit mit graphischen Objekten, die zum jeden angegebenen Chart gehören, bestimmt ist. Diese Funktionen können nicht in Anzeigern verwendet werden.

Die Funktionen, die die Eigenschaften von grafischen Objekten eingeben, und die Operationen der Erstellung [ObjectCreate\(\)](#) und Bewegen [ObjectMove\(\)](#) von Objekte auf dem Chart sind tatsächlich verwendet, um Befehle dem Chart zu senden. Der erfolgreiche Erfüllung dieser Funktionen wird der Befehl in die allgemeine Warteschlange der Chartereignisse platziert. Visuelle Veränderung der Eigenschaften von grafischen Objekten wird während der Verarbeitung der Ereigniswarteschlange von diesem Chart geändert.

Aus diesem Grund sollte man nicht erwarten, dass die graphische Objekten sofort nach dem Aufruf dieser Funktionen visuell aktualisiert wird. Im Allgemeinen werden die graphische Objekten durch das Terminal automatisch nach Änderung-Ereignisse aktualisiert - die Ankunft der neuen Quote, Änderung von Chartfenstergröße, etc.

Um graphischen Objekten zu aktualisieren, verwenden Sie [ChartRedraw\(\)](#).

| Funktion                             | Massnahme                                                                                                          |
|--------------------------------------|--------------------------------------------------------------------------------------------------------------------|
| <a href="#">ObjectCreate</a>         | Erzeugt das Objekt des vorgegebenen Typs auf dem vorgegebenen Chart                                                |
| <a href="#">ObjectName</a>           | Gibt den Namen des Objekts des entsprechenden Typs im angegebenen Chart zurück (im angegebenen Fenster des Charts) |
| <a href="#">ObjectDelete</a>         | Entfernt das Objekt mit dem angegebenen Namen vom angegebenen Chart (vom angegebenen Subfenster des Charts)        |
| <a href="#">ObjectsDeleteAll</a>     | Entfernt alle Objekte des angegebenen Typs vom angegebenen Chart (vom angegebenen Subfenster des Charts)           |
| <a href="#">ObjectFind</a>           | Sucht Objekt mit dem angegebenen Identifikator nach dem Namen                                                      |
| <a href="#">ObjectGetTimeByValue</a> | Gibt Zeitwert für den angegebenen Preiswert des Objekts zurück                                                     |
| <a href="#">ObjectGetValueByTime</a> | Gibt Preiswert des Objekts für die angegebene Zeit zurück                                                          |
| <a href="#">ObjectMove</a>           | Verändert die Koordinaten des angegebenen Bezugspunktes des Objekts.                                               |
| <a href="#">ObjectsTotal</a>         | Gibt die Anzahl der Objekte des angegebenen Typs im angegebenen Chart (angegebenen Subfenster des Charts) zurück.  |
| <a href="#">ObjectGetDouble</a>      | Gibt den Wert des Typs double der entsprechenden Eigenschaft des Objekts zurück.                                   |

|                                         |                                                                                                                                                     |
|-----------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#"><u>ObjectGetInteger</u></a> | Gibt den ganzzahligen Wert der entsprechenden Eigenschaft des Objekts zurück.                                                                       |
| <a href="#"><u>ObjectGetString</u></a>  | Gibt den Wert des Typs string der entsprechenden Eigenschaft des Objekts                                                                            |
| <a href="#"><u>ObjectSetDouble</u></a>  | Stellt den wert der entsprechenden Eigenschaft des Objekts ein                                                                                      |
| <a href="#"><u>ObjectSetInteger</u></a> | Stellt den wert der entsprechenden Eigenschaft des Objekts ein                                                                                      |
| <a href="#"><u>ObjectSetString</u></a>  | Stellt den wert der entsprechenden Eigenschaft des Objekts ein                                                                                      |
| <a href="#"><u>TextSetFont</u></a>      | Setzt die Schrift für die Textausgabe durch Zeichnenmethoden (die Standardschriftart ist Arial 20)                                                  |
| <a href="#"><u>TextOut</u></a>          | Gibt den Text in einem benutzerdefinierten Array (Puffer), der für die Erstellung von grafischen <a href="#"><u>Ressource</u></a> benutzt wird, aus |
| <a href="#"><u>TextGetSize</u></a>      | Gibt die Breite und Höhe der Zeile mit den aktuellen <a href="#"><u>Schrifteinstellungen</u></a> zurück                                             |

Jedes graphische Objekt muss einen Namen haben, der unikal innerhalb eines [Charts](#), sein muss einschliesslich seine Subfenster. Veränderung des Namens des graphischen Objekts formiert zwei Ereignisse: das erste Ereignis ist die Entfernung des Objekts mit dem alten Namen, das zweite Ereignis ist die Erzeugung des graphischen Objekts mit dem neuen Namen.

Nach Erzeugung des Objekt oder Modifikation der [Objekteigenschaften](#) ist es empfehlenswert, die Funktion [ChartRedraw\(\)](#) aufzurufen, die die Anweisung gibt, den Chart (und alle darauf [sichtbaren](#) Objekte) zu zeichnen.

## ObjectCreate

Erzeugt Objekt mit dem angegebenen Namen, Typ und urspruenglichen Koordinaten im angegebenen Subfenster des Charts. Bei der Erzeugung kann man bis auf 30 Koordinaten angeben.

```
bool ObjectCreate(
    long      chart_id,      // Identifikator des Charts
    string     name,         // Objektname
    ENUM_OBJECT type,        // Objekttyp
    int        sub_window,   // Index des Fensters
    datetime   time1,        // Zeit des ersten Bezugspunktes
    double     price1,       // Preis des ersten Bezugspunktes
    ...
    datetime   timeN=0,      // Zeit des N-Bezugspunktes
    double     priceN=0,     // Preis des N-Bezugspunktes
    ...
    datetime   time30=0,     // Zeit 30. Bezugspunktes
    double     price30=0     // Preis des 30. Bezugspunktes
);
```

### Parameter

*chart\_id*

[in] Identifikator des Charts. 0 bedeutet den laufenden Chart.

*name*

[in] Objektname. Name muss unikal im Rahmen eines Charts und seiner Subfenster sein.

*type*

[in] Objekttyp. Wert kann der einer der Aufzählungswerte [ENUM\\_OBJECT](#) sein.

*sub\_window*

[in] Nummer des Subfensters des Charts. Das angegebene Subfenster muss existieren, anderenfalls gibt die Funktion false zurück. .

*time1*

[in] Zeitkoordinate des ersten Bezugspunktes.

*price1*

[in] Preiskoordinate des ersten Bezugspunktes.

*timeN=0*

[in] Zeitkoordinate des N-Bezugspunktes.

*priceN=0*

[in] Preiskoordinate des N-Bezugspunktes.

*time30=0*

[in] Zeikoordinate des 30. Bezugspunktes.

*price30=0*

[in] Preiskoordinate des 30. Bezugspunktes.

**Rückgabewert**

Gibt true oder false zurück, abhängig davon, ob das Objekt erzeugt wurde oder nicht. Für die Erhaltung der zusätzlichen Information über den [Fehler](#), muss die Funktion [GetLastError\(\)](#) aufgerufen werden. Wenn das Objekt davor erzeugt wurde, versucht man, seine Koordinate zu verändern.

**Bemerkung**

Numerierung der Subfenster des Chart (wenn es auf dem Chart Subfenster mit Indikatoren gibt) faengt mit 1 an. Hauptfenster eines Charts gibt es immer und hat Index 0.

Grosse Anzahl der Bezugspunkte (bis 30) ist für weitere Verwendung vorausgesehen. Die Einschraenkung bis 30 Bezugspunkten ist damit verbunden, dass die Anzahl der Parameter beim Funktionsaufruf nicht mehr als 64 sein muss.

Bei der Umbenennung des graphischen Objekts werden gleichzeitig zwei Ereignisse formiert, die im Experten oder im Indikator durch die Funktion [OnChartEvent\(\)](#) verarbeitet werden können :

- Entfernung eines Objektes mit dem alten Namen;
- Erzeugung eines graphischen Objektes mit dem neuen Namen.

Für die Erzeugung jedes der [Objekttypen](#) ist es erforderlich, eine bestimmte Anzahl der Bezugspunkte anzugeben:

| Identifikator                     | Beschreibung                   | Bezugspunkte                                                                           |
|-----------------------------------|--------------------------------|----------------------------------------------------------------------------------------|
| <a href="#">OBJ_VLINE</a>         | Senkrechte Linie               | Ein Bezugspunkt. Tatsächlich wird nur die Koordinate der Zeitachse entlang verwendet.  |
| <a href="#">OBJ_HLINE</a>         | Horizontale Linie              | Ein Bezugspunkt. Tatsächlich wird nur die Koordinate der Preisachse entlang verwendet. |
| <a href="#">OBJ_TREND</a>         | Trendlinie                     | Zwei Bezugspunkte.                                                                     |
| <a href="#">OBJ_TRENDBYANGLE</a>  | Trendlinie nach Winkel         | Zwei Bezugspunkte.                                                                     |
| <a href="#">OBJ_CYCLES</a>        | Zykluslinien                   | Zwei Bezugspunkte.                                                                     |
| <a href="#">OBJ_ARROWED_LINE</a>  | Objekt "Linie mit einem Pfeil" | Zwei Bezugspunkte.                                                                     |
| <a href="#">OBJ_CHANNEL</a>       | Abstandsgleicher Kanal         | Drei Bezugspunkte.                                                                     |
| <a href="#">OBJ_STDDEVCHANNEL</a> | Kanal der Standardabweichung   | Zwei Bezugspunkte.                                                                     |
| <a href="#">OBJ_REGRESSION</a>    | Kanal der linearen Regression  | Zwei Bezugspunkte.                                                                     |
| <a href="#">OBJ_PITCHFORK</a>     | Andrews' Pitchfork             | Drei Bezugspunkte.                                                                     |
| <a href="#">OBJ_GANNLINE</a>      | Gannlinie                      | Zwei Bezugspunkte.                                                                     |
| <a href="#">OBJ_GANNFAN</a>       | Gannfan                        | Zwei Bezugspunkte.                                                                     |
| <a href="#">OBJ_GANNGRID</a>      | Ganngrid                       | Zwei Bezugspunkte.                                                                     |
| <a href="#">OBJ_FIBO</a>          | Fibonacci Retracement          | Zwei Bezugspunkte.                                                                     |



|                                              |                                   |                                                                                                                                        |
|----------------------------------------------|-----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#"><u>OBJ_FIBOTIMES</u></a>         | Zeitzone Fibonacci                | Zwei Bezugspunkte.                                                                                                                     |
| <a href="#"><u>OBJ_FIBOFAN</u></a>           | Fibonacci Fan                     | Zwei Bezugspunkte.                                                                                                                     |
| <a href="#"><u>OBJ_FIBOARC</u></a>           | Fibonacci Arcs                    | Zwei Bezugspunkte.                                                                                                                     |
| <a href="#"><u>OBJ_FIBOCHANNEL</u></a>       | Kanal Fibonacci                   | Drei Bezugspunkte.                                                                                                                     |
| <a href="#"><u>OBJ_EXPANSION</u></a>         | Erweiterung Fibonacci             | Drei Bezugspunkte.                                                                                                                     |
| <a href="#"><u>OBJ_ELLIOTWAVE5</u></a>       | Elliott Motive Wave               | Fünf Bezugspunkte.                                                                                                                     |
| <a href="#"><u>OBJ_ELLIOTWAVE3</u></a>       | Elliott Correction Wave           | Drei Bezugspunkte.                                                                                                                     |
| <a href="#"><u>OBJ_RECTANGLE</u></a>         | Rechteck                          | Zwei Bezugspunkte.                                                                                                                     |
| <a href="#"><u>OBJ_TRIANGLE</u></a>          | Dreieck                           | Drei Bezugspunkte.                                                                                                                     |
| <a href="#"><u>OBJ_ELLIPSE</u></a>           | Ellipse                           | Drei Bezugspunkte.                                                                                                                     |
| <a href="#"><u>OBJ_ARROW_THUMB_UP</u></a>    | Zeichen "Gut" (Daumen oben)       | Ein Bezugspunkt.                                                                                                                       |
| <a href="#"><u>OBJ_ARROW_THUMB_DOWN</u></a>  | Zeichen "Schlecht" (Daumen unten) | Ein Bezugspunkt.                                                                                                                       |
| <a href="#"><u>OBJ_ARROW_UP</u></a>          | Zeichen "Weiche oben"             | Ein Bezugspunkt.                                                                                                                       |
| <a href="#"><u>OBJ_ARROW_DOWN</u></a>        | Zeichen "Weich unten"             | Ein Bezugspunkt.                                                                                                                       |
| <a href="#"><u>OBJ_ARROW_STOP</u></a>        | Zeichen "Stop"                    | Ein Bezugspunkt.                                                                                                                       |
| <a href="#"><u>OBJ_ARROW_CHECK</u></a>       | Zeichen "Kontrollmarke"           | Ein Bezugspunkt.                                                                                                                       |
| <a href="#"><u>OBJ_ARROW_LEFT_PRICE</u></a>  | linke Preismarke                  | Ein Bezugspunkt.                                                                                                                       |
| <a href="#"><u>OBJ_ARROW_RIGHT_PRICE</u></a> | Rechte Preismarke                 | Ein Bezugspunkt.                                                                                                                       |
| <a href="#"><u>OBJ_ARROW_BUY</u></a>         | Zeichen "Buy"                     | Ein Bezugspunkt.                                                                                                                       |
| <a href="#"><u>OBJ_ARROW_SELL</u></a>        | Zeichen "Sell"                    | Ein Bezugspunkt.                                                                                                                       |
| <a href="#"><u>OBJ_ARROW</u></a>             | Objekt "Weiche"                   | Ein Bezugspunkt.                                                                                                                       |
| <a href="#"><u>OBJ_TEXT</u></a>              | Objekt "Text"                     | Ein Bezugspunkt.                                                                                                                       |
| <a href="#"><u>OBJ_LABEL</u></a>             | Objekt "Textmarke"                | Die Position wird durch Eigenschaften <a href="#"><u>OBJPROP_XDISTANCE</u></a> und <a href="#"><u>OBJPROP_YDISTANCE</u></a> angegeben. |
| <a href="#"><u>OBJ_BUTTON</u></a>            | Objekt "Schaltfläche"             | Die Position wird durch Eigenschaften <a href="#"><u>OBJPROP_XDISTANCE</u></a> und <a href="#"><u>OBJPROP_YDISTANCE</u></a> angegeben. |
| <a href="#"><u>OBJ_CHART</u></a>             | Objekt "Chart"                    | Die Position wird durch Eigenschaften <a href="#"><u>OBJPROP_XDISTANCE</u></a> und <a href="#"><u>OBJPROP_YDISTANCE</u></a>            |

|                                            |                                                                                   |                                                                                                                                        |
|--------------------------------------------|-----------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------|
|                                            |                                                                                   | angegeben.                                                                                                                             |
| <a href="#"><u>OBJ_BITMAP</u></a>          | Objekt "Bild"                                                                     | Ein Bezugspunkt.                                                                                                                       |
| <a href="#"><u>OBJ_BITMAP_LABEL</u></a>    | Objekt "graphische Marke"                                                         | Die Position wird durch Eigenschaften <a href="#"><u>OBJPROP_XDISTANCE</u></a> und <a href="#"><u>OBJPROP_YDISTANCE</u></a> angegeben. |
| <a href="#"><u>OBJ_EDIT</u></a>            | Objekt "Eingabefeld"                                                              | Die Position wird durch Eigenschaften <a href="#"><u>OBJPROP_XDISTANCE</u></a> und <a href="#"><u>OBJPROP_YDISTANCE</u></a> angegeben. |
| <a href="#"><u>OBJ_EVENT</u></a>           | Objekt "Ereignis", das einem Ereignis in der wirtschaftlichen Kalender entspricht | Ein Bezugspunkt. Tatsächlich wird nur die Koordinate der Zeitachse entlang verwendet.                                                  |
| <a href="#"><u>OBJ_RECTANGLE_LABEL</u></a> | Objekt "Rechteckige Marke" um eigene grafische Benutzeroberfläche zu entwerfen.   | Die Position wird durch Eigenschaften <a href="#"><u>OBJPROP_XDISTANCE</u></a> und <a href="#"><u>OBJPROP_YDISTANCE</u></a> angegeben. |

## ObjectName

Gibt den Namen des entsprechenden Objekts im angegebenen Chart, angegebenen Subfenster des angegebenen Chart des angegebenen Typs zurück.

```
string ObjectName(  
    long  chart_id,           // Identifikator des Charts  
    int   pos,               // Nummer in der Liste der Objekte  
    int   sub_window=-1,     // Nummer des Fensters  
    int   type=-1            // Typ des Objekts  
);
```

### Parameter

*chart\_id*

[in] Identifikator des Charts. 0 bedeutet den laufenden Chart.

*pos*

[in] Ordnungsnummer des Objekts laut dem angegebenen Filter nach Nummer des Subfensters und Typ.

*sub\_window=-1*

[in] Nummer des Subfensters des Charts, -1 bedeutet alle Subfenster des Charts einschliesslich Hauptfenster.

*type=-1*

[in] Typ des Objekts. Wert kann einer der Aufzählungswerte [ENUM\\_OBJECT](#) sein. -1 bedeutet alle Typen.

### Rückgabewert

Name des Objekts im Erfolgsfall.

### Bemerkung

Bei Umbenennung des graphischen Objekts werden gleichzeitig zwei Ereignisse gebildet, die im Expert oder im Anzeiger durch die Funktion [OnChartEvent\(\)](#) verarbeitet werden können:

- Ereigniss der Entfernung des Objekts mit dem alten Namen;
- Ereignis der Erzeugung des graphischen Objekts mit dem neuen Namen.

## ObjectDelete

Entfernt das Objekt mit dem angegebenen Namen des angegebenen Charts.

```
bool ObjectDelete(  
    long    chart_id,    // Identifikator des Charts  
    string  name         // Name des Objekts  
);
```

### Parameter

*chart\_id*

[in] Identifikator des Charts. 0 bedeutet den laufenden Chart.

*name*

[in] Name des entfernten Objekts.

### Rückgabewert

Gibt true bei der erfolgreichen Entfernung, anderenfalls false. Für die weitere Information über [Fehler](#), muss die Funktion [GetLastError\(\)](#) aufgerufen werden.

### Bemerkung

Bei der Umbenennung des graphischen Objekts werden gleichzeitig zwei Ereignisse gebildet, die im Expert oder im Anzeiger durch die Funktion [OnChartEvent\(\)](#) verarbeitet werden können:

- Ereigniss der Entfernung des Objekts mit dem alten Namen;
- Ereignis der Erzeugung des graphischen Objekts mit dem neuen Namen.

## ObjectsDeleteAll

Entfernt alle Objekte im angegebenen Chart, angegebenen Subfenster des angegebenen Charts des angegebenen Typs.

```
int ObjectsDeleteAll(  
    long  chart_id,           // Identifikator des Charts  
    int   sub_window=-1,     // Index des Fensters  
    int   type=-1            // Typ des Objekts für Entfernung  
);
```

### Parameter

*chart\_id*

[in] Identifikator des Charts. 0 bedeutet den laufenden Chart.

*sub\_window=-1*

[in] Nummer des Subfensters des Charts. 0 bedeutet das Hauptfenster des Charts, -1 bedeutet alle Subfenster des Charts, einschliesslich Hauptfenster.

*type=-1*

[in] Typ des Objekts. Wert kann einer der Aufzählungswerte [ENUM\\_OBJECT](#) sein. -1 bedeutet all Typen.

### Rückgabewert

Gibt die Anzahl der entfernten Objekte zurück. Für die Erhaltung der weiteren Information über den [Fehler](#) muss die Funktion [GetLastError\(\)](#) aufgerufen werden.

## ObjectFind

Sucht ein Objekt mit dem angegebenen Namen auf dem Chart mit dem angegebenen Identifikatoren.

```
int ObjectFind(  
    long    chart_id,    // Identifikator des Charts  
    string  name         // Name des Objekts  
);
```

### Parameter

*chart\_id*

[in] Identifikator des Charts. 0 bedeutet den laufenden Chart.

*name*

[in] Name de gesuchten Objekts.

### Rückgabewert

Im Erfolgsfall gibt die Funktion Nummer des Subfensters zurück (0 bedeutet das Hauptfenster des Chart), in dem sich das gefundene Objekt befindet. Wenn das Objekt nicht gefunden ist, gibt die Funktion eine negative Zahl zurück. Für die Erhaltung der weiteren Information über den [Fehler](#), muss die Funktion [GetLastError\(\)](#) aufgerufen werden.

### Bemerkung

Bei der Umbenennung des graphischen Objekts werden gleichzeitig zwei Ereignisse gebildet, die im Expert oder Anzeiger durch die Funktion [OnChartEvent\(\)](#) verarbeitet werden können:

- Entfernung des Objekts mit dem alten Namen;
- Erzeugung des graphischen Objekts mit dem Namen.

## ObjectGetTimeByValue

Gibt Zeitwert für den angegebenen Preiswert des angegebenen Objekts.

```
datetime ObjectGetTimeByValue(  
    long    chart_id,      // Identifikator des Charts  
    string  name,          // Name des Objekts  
    double  value,         // Preis  
    int     line_id        // Linie  
);
```

### Parameter

*chart\_id*

[in] Identifikator des Charts. 0 bedeutet den laufenden Chart.

*name*

[in] Name des Objekts.

*value*

[in] Preiswert.

*line\_id*

[in] Identifikator der Linie.

### Rückgabewert

Zeitwert für den angegebenen Preiswert des angegebenen Objekts.

### Bemerkung

Da das Objekt in einer Preiskoordinate mehrere Werte haben kann, muss man Nummer der Linie angeben. Diese Funktion ist nur für folgende Objekte anwendbar:

- Trendlinie (OBJ\_TREND)
- Trendlinie nach der Ecke (OBJ\_TRENDBYANGLE)
- Gann Linie (OBJ\_GANNLIN)
- Abstandsgleicher Kanal (OBJ\_CHANNEL) - 2 Linien
- Kanal auf der linealen Regression (OBJ\_REGRESSION) - 3 Linien
- Kanal der Standardabweichung (OBJ\_STDDEVCHANNEL) - 3 Linien
- Linie mit der Weiche (OBJ\_ARROWED\_LINE)

### Sehen Sie auch

[Objekttypen](#)

## ObjectGetValueByTime

Gibt den Preiswert für die angegebene Zeit des angegebenen Objekts.

```
double ObjectGetValueByTime(  
    long      chart_id,      // Identifikator des Charts  
    string     name,         // Objektname  
    datetime  time,         // Zeit  
    int       line_id        // Linie  
);
```

### Parameter

*chart\_id*

[in] Identifikator des Charts. 0 bedeutet den laufenden Chart.

*name*

[in] Objektname.

*time*

[in] Zeitwert.

*line\_id*

[in] Identifikator der Linie.

### Rückgabewert

Preiswert für die angegebene Zeit des angegebenen Objekts.

### Bemerkung

Da das Objekt in einer Preiskordinate mehrere Werte haben kann, muss man die Nummer der Linie angeben. Diese Funktion ist nur für folgende Objekte anwendbar:

- Trendlinie (OBJ\_TREND)
- Trendlinie nach der Ecke (OBJ\_TRENDBYANGLE)
- Gann Linie (OBJ\_GANNLIN)
- Abstandsgleicher Kanal (OBJ\_CHANNEL) - 2 Linien
- Kanal auf der linealen Regression (OBJ\_REGRESSION) - 3 Linien
- Kanal der Standardabweichung (OBJ\_STDDEVCHANNEL) - 3 Linien
- Linie mit der Weiche (OBJ\_ARROWED\_LINE)

### Sehen Sie auch

[Objekttypen](#)



## ObjectMove

Verändert Koordinaten des angegebenen Bezugspunktes des Objekts.

```
bool ObjectMove(  
    long      chart_id,      // Identifikator des Charts  
    string     name,         // Objektname  
    int        point_index,   // Nummer des Bezugspunktes  
    datetime   time,         // Zeit  
    double     price         // Preis  
);
```

### Parameter

*chart\_id*

[in] Identifikator des Charts. 0 bedeutet den laufenden Chart.

*name*

[in] Objektname.

*point\_index*

[in] Nummer des Bezugspunktes. Anzahl der Bezugspunkte hängt vom Objekttyp ab.

*time*

[in] Zeitkoordinate des angegebenen Bezugspunktes.

*price*

[in] Preiskoordinate des angegebenen Bezugspunktes.

### Rückgabewert

Im Erfolgsfall gibt true zurück, anderenfalls false. Für die Erhaltung der weiteren Information über den [Fehler](#), muss die Funktion [GetLastError\(\)](#) aufgerufen werden.

## ObjectsTotal

Gibt die Anzahl der Objekte im angegebenen Chart, angegebenen Subfenster des angegebenen Charts des angegebenen Typs zurück.

```
int ObjectsTotal(  
    long  chart_id,           // Identifikator des Charts  
    int   sub_window=-1,     // Index des Fensters  
    int   type=-1            // Objekttyp  
);
```

### Parameter

*chart\_id*

[in] Identifikator des Charts. 0 bedeutet den laufenden Chart.

*sub\_window=-1*

[in] Nummer des Subfensters des Charts. 0 bedeutet Hauptfenster des Charts, -1 bedeutet alle Subfenster des Charts, einschliesslich Hauptfenster.

*type=-1*

[in] Objekttyp. Wert kann einer der Aufzählungswerte [ENUM\\_OBJECT](#) sein. -1 bedeutet alle Typen.

### Rückgabewert

Anzahl der Objekte.



```

void OnStart()
{
//--- Hilfsfelder
double high[],low[],price1,price2;
datetime time[],time1,time2;
//--- kopieren wir Eroeffnungspreise - 100 letzte Bars reicht es
int copied=CopyHigh(Symbol(),0,0,100,high);
if(copied<=0)
{
Print("Nicht gelungen, Werte der Preisserie zu kopieren High");
return;
};
//--- kopieren wir Schlusspreise - 100 letzte Bars reicht es
copied=CopyLow(Symbol(),0,0,100,low);
if(copied<=0)
{
Print("Misslugen, die Werte der Preisserie Low" zu kopieren);
return;
};
//--- kopieren wir Eroeffnungszeit für 100 letzte Bars
copied=CopyTime(Symbol(),0,0,100,time);
if(copied<=0)
{
Print("Misslungen, die Werte der Preisserie Time zu kopieren");
return;
};
//--- organisieren wir Zugang zu kopierten Daten als zu einer Preisserie - umgekehrt
ArraySetAsSeries(high,true);
ArraySetAsSeries(low,true);
ArraySetAsSeries(time,true);

//--- Koordinaten des ersten Bezugspunktes des Fibo-Objektes
price1=high[70];
time1=time[70];
//--- Koordinaten des ersten Bezugspunktes des Fiboobjektes
price2=low[50];
time2=time[50];

//--- nun muessen wir ein Fibo-Objekt selbst erzeugen
bool created=ObjectCreate(0,"Fibo",OBJ_FIBO,0,time1,price1,time2,price2);
if(created) // wenn das Objekt erfolgreich erzeugt wurde
{
//--- geben wir die Farbe der Fibo-Levels vor
ObjectSetInteger(0,"Fibo",OBJPROP_LEVELCOLOR,Blue);
//--- uebrigens, wieviel Fibo-Levels haben wir ?
int levels=ObjectGetInteger(0,"Fibo",OBJPROP_LEVELS);
Print("Fibo levels before = ",levels);
//---geben wir in Journal=> Levelnummer:Werte Beschreibung_des Levels
for(int i=0;i<levels;i++)

```

```

{
    Print(i,":",ObjectGetDouble(0,"Fibo",OBJPROP_LEVELVALUE,i),
          " ",ObjectGetString(0,"Fibo",OBJPROP_LEVELTEXT,i));
};
//--- versuchen wir, die Anzahl der Levels um 1 zu vergroessern
bool modified=ObjectSetInteger(0,"Fibo",OBJPROP_LEVELS,levels+1);
if(!modified) // Mislungen, die Anzahl der Levels zu verändern
{
    Print("Mislungen die Anzahl der Levels Fibo zu verändern, Fehler ",GetLastError());
};
//--- wir melden nur
Print("Fibo levels after = ",ObjectGetInteger(0,"Fibo",OBJPROP_LEVELS));
//--- geben wir den Wert für den nur erzeugten Level vor
bool added=ObjectSetDouble(0,"Fibo",OBJPROP_LEVELVALUE,levels,133);
if(added) // gelungen, Wert für Level vorzugeben
{
    Print("Gelungen, noch einen Level Fibo einzustellen");
    //--- nicht vergessen, Levelbeschreibung vorzugeben
    ObjectSetString(0,"Fibo",OBJPROP_LEVELTEXT,levels,"my level");
    ChartRedraw(0);
    //--- bekommen wir aktuelle Werte der Levelsanzahl in Fibo-Objekt
    levels=ObjectGetInteger(0,"Fibo",OBJPROP_LEVELS);
    Print("Fibo levels after adding = ",levels);
    //--- geben wir nochmals alle Levels aus - nur um uns zu versichern
    for(int i=0;i<levels;i++)
    {
        Print(i,":",ObjectGetDouble(0,"Fibo",OBJPROP_LEVELVALUE,i),
              " ",ObjectGetString(0,"Fibo",OBJPROP_LEVELTEXT,i));
    };
};
else // Misserfolg beim Versuch Levelsanzahl in einem Fibo-Objekt zu vergroessen
{
    Print("Mislungen, noch einen Level Fibo einzustellen. Fehler ",GetLastError());
};
};
};

```

Sehen Sie auch

[Objekttypen](#), [Objekteigenschaften](#)

## ObjectSetInteger

Gibt den Wert der entsprechenden Eigenschaft des Objekts vor. Eigenschaft des Objekts muss der Typen [datetime](#), [int](#), [color](#), [bool](#) oder [char](#) sein. Es gibt 2 Varianten der Funktion.

### Einstellung des Wertes der Eigenschaft ohne Modifikator

```
bool ObjectSetInteger(
    long    chart_id,          // Identifikator des Charts
    string  name,              // Name
    int     prop_id,           // Eigenschaft
    long    prop_value         // Wert
);
```

### Einstellung des Wertes der Eigenschaft mit Andeutung des Modifikatoren

```
bool ObjectSetInteger(
    long    chart_id,          // Identifikator des Diagramms
    string  name,              // Name
    int     prop_id,           // Eigenschaft
    int     prop_modifier,     // Modifikator
    long    prop_value         // Wert
);
```

### Parameter

*chart\_id*

[in] Identifikator des Charts. 0 bedeutet den laufenden Chart.

*name*

[in] Objektname.

*prop\_id*

[in] Identifikator der Eigenschaft des Objekts. Wert kann einer der Aufzählungswerte [ENUM\\_OBJECT\\_PROPERTY\\_INTEGER](#) sein.

*prop\_modifier*

[in] Modifikator der angegebenen Eigenschaft. Die meisten Eigenschaften erfordern keinen Modifikator.

*prop\_value*

[in] Wert der Eigenschaft.

### Rückgabewert

Die Funktion gibt true zurück nur wenn der Befehl zum Ändern der Eigenschaften von Objekten zum Chart erfolgreich gesendet wurde. Ansonsten gibt es false zurück. Für die Erhaltung der weiteren Information über den [Fehler](#), muss die Funktion [GetLastError\(\)](#) aufgerufen werden.

### Beispiel der Erstellung einer [Web-Farbtabelle](#)

```

//+-----+
//|                                     Table of Web Colors|
//|                                     Copyright 2011, MetaQuotes Software Corp |
//|                                     http://www.metaquotes.net |
//+-----+
#define X_SIZE 140          // Breite des Objekts OBJ_EDIT
#define Y_SIZE 33          // Breite des Objekts OBJ_EDIT
//+-----+
//| Ein Array von Web-Farben |
//+-----+
color ExtClr[140]=
{
    clrAliceBlue,clrAntiqueWhite,clrAqua,clrAquamarine,clrAzure,clrBeige,clrBisque,clr
    clrBlue,clrBlueViolet,clrBrown,clrBurlyWood,clrCadetBlue,clrChartreuse,clrChocolate
    clrCornsilk,clrCrimson,clrCyan,clrDarkBlue,clrDarkCyan,clrDarkGoldenrod,clrDarkGray
    clrDarkMagenta,clrDarkOliveGreen,clrDarkOrange,clrDarkOrchid,clrDarkRed,clrDarkSalmon
    clrDarkSlateBlue,clrDarkSlateGray,clrDarkTurquoise,clrDarkViolet,clrDeepPink,clrDeep
    clrDodgerBlue,clrFireBrick,clrFloralWhite,clrForestGreen,clrFuchsia,clrGainsboro,clr
    clrGoldenrod,clrGray,clrGreen,clrGreenYellow,clrHoneydew,clrHotPink,clrIndianRed,clr
    clrLavender,clrLavenderBlush,clrLawnGreen,clrLemonChiffon,clrLightBlue,clrLightCoral
    clrLightGoldenrod,clrLightGreen,clrLightGray,clrLightPink,clrLightSalmon,clrLightSea
    clrLightSlateGray,clrLightSteelBlue,clrLightYellow,clrLime,clrLimeGreen,clrLinen,clr
    clrMediumAquamarine,clrMediumBlue,clrMediumOrchid,clrMediumPurple,clrMediumSeaGreen
    clrMediumSpringGreen,clrMediumTurquoise,clrMediumVioletRed,clrMidnightBlue,clrMintC
    clrNavajoWhite,clrNavy,clrOldLace,clrOlive,clrOliveDrab,clrOrange,clrOrangeRed,clrOr
    clrPaleGreen,clrPaleTurquoise,clrPaleVioletRed,clrPapayaWhip,clrPeachPuff,clrPeru,cl
    clrPurple,clrRed,clrRosyBrown,clrRoyalBlue,clrSaddleBrown,clrSalmon,clrSandyBrown,cl
    clrSienna,clrSilver,clrSkyBlue,clrSlateBlue,clrSlateGray,clrSnow,clrSpringGreen,cl
    clrThistle,clrTomato,clrTurquoise,clrViolet,clrWheat,clrWhite,clrWhiteSmoke,clrYell
};
//+-----+
//| Erstellung und Initialisierung des Objekts OBJ_EDIT |
//+-----+
void CreateColorBox(int x,int y,color c)
{
    //--- Wir generieren einen Namen für das neue Objekt auf den Farbnamen
    string name="ColorBox_"+(string)x+"_"+(string)y;
    //--- Wir erstellen ein neues Objekt OBJ_EDIT
    if(!ObjectCreate(0,name,OBJ_EDIT,0,0,0))
    {
        Print("Konnte nicht erstellen '",name,'"");
        return;
    }
    //--- Wir definieren die Koordinate des Ankerpunktes, Breite und Höhe in Pixel
    ObjectSetInteger(0,name,OBJPROP_XDISTANCE,x*X_SIZE);
    ObjectSetInteger(0,name,OBJPROP_YDISTANCE,y*Y_SIZE);
    ObjectSetInteger(0,name,OBJPROP_XSIZE,X_SIZE);
    ObjectSetInteger(0,name,OBJPROP_YSIZE,Y_SIZE);
    //--- Wir geben die Textfarbe für das Objekt ein
    if(clrBlack==c) ObjectSetInteger(0,name,OBJPROP_COLOR,clrWhite);
    else ObjectSetInteger(0,name,OBJPROP_COLOR,clrBlack);
    //--- Wir geben die Hintergrundfarbe ein
    ObjectSetInteger(0,name,OBJPROP_BGCOLOR,c);
    //--- Wir setzen den Text des Objekts OBJ_EDIT entsprechend der Hintergrundfarbe
    ObjectSetString(0,name,OBJPROP_TEXT,(string)c);
}
//+-----+
//| Die Funktion starter Skript |
//+-----+
void OnStart()
{

```

```
//--- Wir erstellen eine Tabelle der Farbblöcke 7x20
for(uint i=0;i<140;i++)
    CreateColorBox(i%7,i/7,ExtClr[i]);
}
```

Sehen Sie auch

[Objekttypen](#), [Objekteigenschaften](#)



## ObjectSetString

Gibt den Wert der entsprechenden Eigenschaft des Objekts. Eigenschaft muss des Typs [string](#) sein. Es gibt 2 Varianten der Funktion.

### Einstellung des Wertes der Eigenschaft ohne Modifikator

```
bool ObjectSetString(
    long    chart_id,      // Identifikator des Charts
    string  name,          // Name
    int     prop_id,       // Eigenschaft
    string  prop_value     // Wert
);
```

### Einstellung des Wertes der Eigenschaft mit Andeutung von Modifikator

```
bool ObjectSetString(
    long    chart_id,      // Identifikator des Charts
    string  name,          // Name
    int     prop_id,       // Eigenschaft
    int     prop_modifier, // Modifikator
    string  prop_value     // Wert
);
```

### Parameter

*chart\_id*

[in] Identifikator des Charts. 0 bedeutet den laufenden Chart.

*name*

[in] Objektname.

*prop\_id*

[in] Identifikator der Eigenschaft des Objekts. Wert kann einer der Aufzählungswerte [ENUM\\_OBJECT\\_PROPERTY\\_STRING](#) sein.

*prop\_modifier*

[in] Modifikator der angegebenen Eigenschaft. Die meisten Eigenschaftens erfordern keinen Modifikator. Bedeutet die Standnummer in [Fibonacci\\_Werkzeuge](#) und in Andrew's Pitchfork. Nummerierung beginnt bei Null.

*prop\_value*

[in] Wert der Eigenschaft.

### Rückgabewert

Die Funktion gibt true zurück nur wenn der Befehl zum Ändern der Eigenschaften von Objekten zum Chart erfolgreich gesendet wurde. Ansonsten gibt es false zurück. Für die Erhaltung der weiteren Information über den [Fehler](#), muss die Funktion [GetLastError\(\)](#) aufgerufen werden.

### Bemerkung

Bei der Umbenennung des graphischen Objekts werden gleichzeitig zwei Ereignisse gebildet, die

durch die Funktion [OnChartEvent\(\)](#) im Expert oder Anzeiger verarbeitet werden können:

- Entfernung des Objekts mit dem alten Namen;
- Erzeugung des graphischen Objekts mit dem neuen Namen.

## ObjectGetDouble

Gibt den Wert der entsprechenden Eigenschaft des Objekt zurück. Eigenschaft des Objekts muss des Typs double sein. Es gibt 2 Varianten der Funktion.

1. Gibt den Wert der Eigenschaft sofort zurück.

```
double ObjectGetDouble(
    long    chart_id,          // Identifikator des Charts
    string  name,              // Objektname
    int     prop_id,           // Identifikator der Eigenschaft
    int     prop_modifier=0    // Modifikator der Eigenschaft wenn erforderlich
);
```

2. Gibt true oder false zurück abhängig davon, ob die Funktion erfolgreich durchgeführt wird. Im Erfolgsfall wird die Funktion in eine Empfangsvariable gesetzt, die durch Referenz vom letzten Parameter übertragen wird.

```
bool ObjectGetDouble(
    long    chart_id,          // Identifikator des Charts
    string  name,              // Objektname
    int     prop_id,           // Identifikator der Eigenschaft
    int     prop_modifier,     // Modifikator der Eigenschaft
    double& double_var         // hier erhalten wir den Wert der Eigenschaft
);
```

### Parameter

*chart\_id*

[in] Identifikator des Charts. 0 bedeutet den laufenden Chart.

*name*

[in] Objektname.

*prop\_id*

[in] Identifikator der Eigenschaft des Objekts. Wert kann einer der Aufzählungswerte [ENUM\\_OBJECT\\_PROPERTY\\_DOUBLE](#) sein.

*prop\_modifier*

[in] Modifikator der angegebenen Eigenschaft. Für die erste Variante des Aufrufs ist der Default-Wert des Modifikators 0. Die meisten Eigenschaften erfordern keinen Modifikator. Bedeutet die Standnummer in [Fibonacci Werkzeuge](#) und in Andrew's Pitchfork. Nummerierung beginnt bei Null.

*double\_var*

[out] Variable des Typs double, die den Wert der angeforderten Eigenschaft annimmt.

### Rückgabewert

Wert des Typs double für die erste Variante des Aufrufes.

für die zweite Variante des Aufrufs gibt true zurück, wenn diese Eigenschaft unterstützt wird und der Wert in die Variable double\_var gesetzt wurde, anderenfalls gibt false zurück. Für Erhaltung der weiteren Information über den [Fehler](#), muss die Funktion [GetLastError\(\)](#) aufgerufen werden.

## ObjectGetInteger

Gibt den Wert der entsprechenden Eigenschaft des Objekts zurück. Eigenschaft des Objekts muss der Typen datetime, int, color, bool oder char sein. Es gibt 2 Varianten der Funktion.

1. Gibt den Wert der Eigenschaft direkt zurück.

```
long ObjectGetInteger(
    long    chart_id,          // Identifikator des Charts
    string  name,              // Objektname
    int     prop_id,           // Identifikator der Eigenschaft
    int     prop_modifier=0    // Modifikator der Eigenschaft wenn erforderlich
);
```

2. Gibt true oder false zurück abhängig davon, ob die Funktion erfolgreich durchgeführt wird. Im Erfolgsfall wird der Wert der Eigenschaft in die Empfangsvariable gesetzt, die durch Referenz vom letzten Parameter übertragen wird.

```
bool ObjectGetInteger(
    long    chart_id,          // Identifikator des Charts
    string  name,              // Objektname
    int     prop_id,           // Identifikator der Eigenschaft
    int     prop_modifier,     // Modifikator der Eigenschaft
    long&   long_var           // hier erhalten wir den Wert der Eigenschaft
);
```

### Parameter

*chart\_id*

[in] Identifikator des Charts. 0 bedeutet den laufenden Chart.

*name*

[in] Objektname.

*prop\_id*

[in] Identifikator der Eigenschaft des Objekts. Wert kann einer der Aufzählungswerte ENUM\_OBJECT\_PROPERTY\_INTEGER sein.

*prop\_modifier*

[in] Modifikator der angegebenen Eigenschaft. Für die erste Variante ist der Default-Wert des Modifikators 0. Die meisten Eigenschaften erfordern keinen Modifikator. Bedeutet die Standnummer in Fibonacci Werkzeuge und in Andrew's Pitchfork. Nummerierung beginnt bei Null.

*long\_var*

[out] Variable des Typs long, die den Wert der angeforderten Eigenschaft annimmt.

### Rückgabewert

Wert des Typs long für die erste Variante des Aufrufs.

für die zweite Variante des Aufrufs gibt true zurück, wenn diese Eigenschaft unterstützt wird und der Wert in die Variable long\_var gesetzt wurde, anderenfalls gibt false zurück. Für Erhaltung der weiteren Information über den Fehler, muss die Funktion GetLastError() aufrufen werden.

## ObjectGetString

Gibt den Wert der entsprechenden Eigenschaft des Objekts zurück. Eigenschaft des Objekts muss des Typs string sein. Es gibt 2 Varianten der Funktion.

1. Gibt den wert der Eigenschaft sofort zurück.

```
string ObjectGetString(
    long    chart_id,          // Identifikator des Charts
    string  name,              // Objektname
    int     prop_id,           // Identifikator der Eigenschaft
    int     prop_modifier=0    // Modifikator der Eigenschaft, wenn erforderlich
);
```

2. Gibt true oder false zurück abhängig davon, ob die Funktion erfolgreich durchgeführt wurde. Im Erfolgsfall wird der Wert der Eigenschaft in die Empfangsvariable gesetzt, die durch Referenz vom letzten Parameter übertragen wird.

```
bool ObjectGetString(
    long    chart_id,          // Identifikator des Charts
    string  name,              // Objektname
    int     prop_id,           // Identifikator der Eigenschaft
    int     prop_modifier,     // Modifikator der Eigenschaft
    string& string_var         // hier erhalten wir den Wert der Eigenschaft
);
```

### Parameter

*chart\_id*

[in] Identifikator des Charts. 0 bedeutet den laufenden Chart.

*name*

[in] Objektname.

*prop\_id*

[in] Identifikator der Eigenschaft des Objekts. Wert kann einer der Aufzählungswerte ENUM\_OBJECT\_PROPERTY\_STRING sein.

*prop\_modifier*

[in] Modifikator der angegebenen Eigenschaft. Für die erste Default-Variante ist der Wert des Modifikators gleich 0. Die meisten Eigenschaften erfordern keinen Modifikator. It denotes the number of the level in Fibonacci tools and in the graphical object Andrew's pitchfork. The numeration of levels starts from zero.

*string\_var*

[out] Variable des Typs string, die den Wert der angeforderten Eigenschaft annimmt.

### Rückgabewert

Wert des Typs string für die erste Variante des Aufrufes.

für die zweite Variante des Aufrufs gibt true zurück, wenn diese Eigenschaft unterstuetzt wird und der Wert in die Variable string\_var gesetzt wurde, anderenfalls gibt false zurück. Für Erhaltung der

weitere Information über den [Fehler](#), muss die Funktion [GetLastError\(\)](#) aufgerufen werden.

**Bemerkung**

Bei der Umbenennung des graphischen Objekts werden gleichzeitig zwei Ereignisse gebildet, die im Expert oder Anzeiger durch die Funktion [OnChartEvent\(\)](#) verarbeitet können:

- Entfernung des Objekts mit dem alten Namen;
- Erzeugung des graphischen Objekts mit dem neuen Namen.

## TextSetFont

Setzt die Schrift für die Textausgabe durch Zeichenmethoden und dann gibt das Ergebnis der Erfolg dieser Operation zurück. Die Standardschriftart ist Arial -120 (12 pt).

```
bool TextSetFont(  
    const string name,           // Schriftname und Pfad zur Schriftdatei auf der Festplatte  
    int size,                   // Schriftgröße  
    uint flags,                 // Kombination von Flags  
    int orientation=0           // Winkel des Textes  
);
```

### Optionen

*name*

[in] Schriftname und Pfad zur Schriftdatei auf der Festplatte.

*size*

[in] Die Schriftgröße kann als positive und negative Werte eingegeben werden. Bei positiven Werten ist die Größe des angegebenen Textes nicht auf Schriftgrößeneinstellungen im Betriebssystem abhängen. Bei negativen Werten wird der Wert in Zehntelpunkt angegeben, und Textgröße wird von den Systemeinstellungen ("Standardmaßstab" oder "großen Maßstab") abhängen. Weitere Informationen über die Unterschiede in den Betriebsarten finden Sie in der Note.

*flags*

[in] Kombination der [Flaggen](#), die den Stil der Schriftart beschreiben.

*orientation*

[in] Horizontaler Winkel des Textes zu der Achse X, Maßeinheit ist 0,1 Grad. Das heißt *orientation=450* bedeutet Neigung von 45 Grad.

### Rückgabewert

Gibt true zurück, wenn die aktuelle Schrift erfolgreich gesetzt wird, ansonsten false. Mögliche Fehlercodes:

- `ERR_INVALID_PARAMETER(4003)` - *name* ist NULL oder "" (leere Zeile),
- `ERR_INTERNAL_ERROR(4001)` - Betriebssystemfehler (zum Beispiel der Versuch, eine nicht vorhandene Schriftart zu erstellen).

### Hinweis

Wenn ":" im Schriftname verwendet wird, dann wird die Schriftart aus der [EX5-Ressource](#) heruntergeladen. Wenn der Schriftname *name* mit der Erweiterung angegeben ist, wird die Schriftart aus einer Datei geladen, in diesem Fall - wenn der Pfad mit "\" oder "/" beginnt, wird die Datei relativ zu dem Verzeichnis MQL5 gesucht, ansonsten wird sie relativ zu dem Pfad der EX5-Datei, die die Funktion `TextSetFont()` anruft hat, gesucht.

Die Schriftgröße wird als positive oder negative Werte angegeben, definiert das Zeichen die Abhängigkeit der Textgröße von den Betriebssystemeinstellungen (Schriftskala).

- Wenn die Größe durch eine positive Zahl angegeben ist, so während der Anzeige der logischen Schriftart als eine physikalische Schriftart, wird die Größe in der physikalischen Maßeinheit des Gerätes (Pixel) umgewandelt, und diese Größe entspricht der Höhe der Zeichenzellen aus

verfügbaren Schriftarten. Es ist nicht empfohlen in den Fällen, wenn die gemeinsame Nutzung der durch Funktion [TextOut\(\)](#) ausgegebenen Texten und durch das Graphikobjekt [OBJ\\_LABEL](#) ("Text-Label") angezeigten Texten auf einem Chart erwartet ist.

- Wenn die Größe durch eine negative Zahl angegeben ist, so ist die angegebene Größe in Zehntelverknüpfungspunkt angegeben (Wert -350 bedeutet 35 logischen Punkte) und wird dividiert durch 10, und dann wird der resultierende Wert in die physikalischen Geräteeinheiten (Pixel) umgewandelt und entspricht den absoluten Wert der Symbolhöhe von verfügbaren Schriftarten. Um den Text auf dem Bildschirm mit der gleichen Größe wie im Objekt [OBJ\\_LABEL](#) zu erhalten, nehmen Sie die Schriftgröße, die in Objekt-Eigenschaften angegeben ist, und multiplizieren Sie sie mit -10.

Die Flaggen können als eine Kombination von Stil-Flaggen mit eine der Flaggen, die die Dicke der Schrift angeben, verwendet werden. Namen der Flaggen sind unten verfügbar.

#### Flaggen, um Schriftstil anzugeben

| Flagge         | Beschreibung   |
|----------------|----------------|
| FONT_ITALIC    | Kursive        |
| FONT_UNDERLINE | Unterstreichen |
| FONT_STRIKEOUT | Durchstreichen |

#### Flaggen, um die Dicke der Schriftart anzugeben

| Flagge        |
|---------------|
| FW_DONTCARE   |
| FW_THIN       |
| FW_EXTRALIGHT |
| FW_ULTRALIGHT |
| FW_LIGHT      |
| FW_NORMAL     |
| FW_REGULAR    |
| FW_MEDIUM     |
| FW_SEMIBOLD   |
| FW_DEMIBOLD   |
| FW_BOLD       |
| FW_EXTRABOLD  |
| FW_ULTRABOLD  |
| FW_HEAVY      |
| FW_BLACK      |

Sehen Sie auch



Ressourcen, ResourceCreate(), ResourceSave(), TextOut()

## TextOut

Gibt den Text in einem benutzerdefinierten Array (Puffer) aus und dann gibt das Ergebnis der Erfolg dieser Operation zurück. Dieses Array wird für die Erstellung von der grafischen [Ressource](#) verwendet.

```
bool TextOut(
    const string      text,           // Textausgabe
    int               x,              // X Koordinate
    int               y,              // Y Koordinate
    uint              anchor,         // Bindungsmethode
    uint               &data[],       // Puffer für Ausgabe
    uint              width,          // Pufferbreite in Punkte
    uint              height,         // Pufferhöhe in Punkte
    uint              color,          // Textfarbe
    ENUM_COLOR_FORMAT color_format   // Farbformat für Ausgabe
);
```

### Optionen

*text*

[in] Der Text, der in den Puffer geschrieben wird. Nur einzeilige Textausgabe verfügbar ist.

*x*

[in] X Koordinate des Ankerpunktes des ausgegebenen Textes.

*y*

[in] Y Koordinate des Ankerpunktes des ausgegebenen Textes.

*anchor*

[In] Der Wert aus der Gruppe von neun vordefinierten Methoden der Position des Ankerpunktes des angezeigten Textes. Wird durch eine Kombination von zwei Flaggen gesetzt - die Flagge der horizontalen Textausrichtung und die Flagge der vertikalen Textausrichtung. Namen der Flaggen sind in Bemerkung verfügbar.

*data[]*

[in] Der Puffer, in dem der Text ausgegeben wird. Dieser Puffer wird für die Erstellung von der grafischen [Ressource](#) verwendet.

*width*

[in] Die Breite des Puffers in Punkte (Pixels).

*height*

[in] Die Höhe des Puffers in Punkte (Pixels).

*color*

[in] Textfarbe.

*color\_format*

[in] Farbformat wird durch einen Wert aus der Aufzählung [ENUM\\_COLOR\\_FORMAT](#) angegeben.

### Rückgabewert

Gibt bei Erfolg true zurück, ansonsten false.

## Hinweis

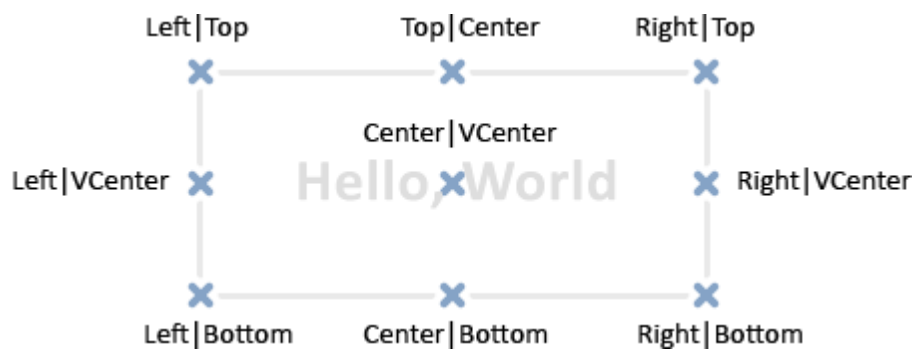
Die Bindungsmethode, die durch Parameter *anchor* angegeben wird, ist eine Kombination von zwei Flaggen von der vertikalen und horizontalen Textausrichtung. Flagge der horizontalen Textausrichtung:

- TA\_LEFT - Ankerpunkt auf der linken Seite des Begrenzungsrahmens
- TA\_CENTER - Ankerpunkt ist horizontal in der Mitte des Begrenzungsrahmens
- TA\_RIGHT - Ankerpunkt auf der rechten Seite des Begrenzungsrahmens

Flagge der vertikalen Textausrichtung:

- TA\_TOP - Ankerpunkt auf der oberen Seite des Begrenzungsrahmens
- TA\_VCENTER - Ankerpunkt ist vertikal in der Mitte des Begrenzungsrahmens
- TA\_BOTTOM - Ankerpunkt auf der unteren Seite des Begrenzungsrahmens

Mögliche Kombinationen von Flaggen und die Bindungsmethoden, die durch die Flaggen gesetzt werden, sind im Bild angezeigt.



## Beispiel:

```
//--- Die Breite und Höhe der Leinwand (Leinwand, auf der wir zeichnen)
#define IMG_WIDTH 200
#define IMG_HEIGHT 200
//--- bevor das Skript zu starten, zeigen wir ein Fenster mit Parameter
#property script_show_inputs
//--- Möglichkeit Farbe anzugeben
input ENUM_COLOR_FORMAT clr_format=COLOR_FORMAT_XRGB_NOALPHA;
//--- Array (Puffer) zu zeichnen
uint ExtImg[IMG_WIDTH*IMG_HEIGHT];
//+-----+
//| Script program start function |
//+-----+
void OnStart()
{
    //--- wir erstellen das Objekt OBJ_BITMAP_LABEL zu zeichnen
    ObjectCreate(0, "CLOCK", OBJ_BITMAP_LABEL, 0, 0, 0);
    //--- wir geben den Namen der Ressource für Zeichnung im Objekt CLOCK an
    ObjectSetString(0, "CLOCK", OBJPROP_BMPFILE, "::IMG");

    //--- Hilfsvariable
    double a;           // Winkel des Pfeils
```

```

uint   nm=2700;           // Minutenzähler
uint   nh=2700*12;        // Stundenzähler
uint   w,h;               // Variable für Größe der Textzeilen
int     x,y;               // Variable für Berechnung der aktuellen Ankerpunktkoordinate

//--- Wir drehen die Uhr in einer Endlosschleife, da das Skript nicht aufgehört wird
while(!IsStopped())
{
    //--- Inhalt des Arrays von Uhr-Zeichnung Puffer löschen
    ArrayFill(ExtImg,0,IMG_WIDTH*IMG_HEIGHT,0);
    //--- Schriftart für Zahlen auf dem Zifferblatt angeben
    TextSetFont("Arial",-200,FW_EXTRABOLD,0);
    //--- Zifferblatt zeichnen
    for(int i=1;i<=12;i++)
    {
        //--- Größe der aktuellen Stunde auf dem Zifferblatt erhalten
        TextGetSize(string(i),w,h);
        //--- Koordinaten der aktuellen Stunde auf dem Zifferblatt berechnen
        a=-((i*300)%3600*M_PI)/1800.0;
        x=IMG_WIDTH/2-int(sin(a)*80+0.5+w/2);
        y=IMG_HEIGHT/2-int(cos(a)*80+0.5+h/2);
        //--- diese Stunde auf dem Zifferblatt in Puffer ExtImg[] ausgeben
        TextOut(string(i),x,y,TA_LEFT|TA_TOP,ExtImg,IMG_WIDTH,IMG_HEIGHT,0xFFFFFFFF,clr_fo
    }
    //--- Jetzt geben wir eine Schriftart für den Minutenzeiger an
    TextSetFont("Arial",-200,FW_EXTRABOLD,-int(nm%3600));
    //--- Größe des Minutenzeigers erhalten
    TextGetSize("----->",w,h);
    //--- Koordinaten des Minutenzeigers auf dem Zifferblatt berechnen
    a=-(nm%3600*M_PI)/1800.0;
    x=IMG_WIDTH/2-int(sin(a)*h/2+0.5);
    y=IMG_HEIGHT/2-int(cos(a)*h/2+0.5);
    //--- Minutenzeiger auf dem Zifferblatt in Puffer ExtImg[] ausgeben
    TextOut("----->",x,y,TA_LEFT|TA_TOP,ExtImg,IMG_WIDTH,IMG_HEIGHT,0xFFFFFFFF,clr_fo

    //--- Jetzt geben wir eine Schriftart für den Stundenzeiger an
    TextSetFont("Arial",-200,FW_EXTRABOLD,-int(nh/12%3600));
    TextGetSize("==>",w,h);
    //--- Koordinaten des Stundenzeigers auf dem Zifferblatt berechnen
    a=-(nh/12%3600*M_PI)/1800.0;
    x=IMG_WIDTH/2-int(sin(a)*h/2+0.5);
    y=IMG_HEIGHT/2-int(cos(a)*h/2+0.5);
    //--- Stundenzeiger auf dem Zifferblatt in Puffer ExtImg[] ausgeben
    TextOut("==>",x,y,TA_LEFT|TA_TOP,ExtImg,IMG_WIDTH,IMG_HEIGHT,0xFFFFFFFF,clr_for

    //--- Grafikressource aktualisieren
    ResourceCreate("::IMG",ExtImg,IMG_WIDTH,IMG_HEIGHT,0,0,IMG_WIDTH,clr_format);
    //--- Zwangsaktualisierung des Charts

```

```
ChartRedraw();

//--- Die Erhöhung der Stunden- und Minutenzähler
nm+=60;
nh+=60;
//--- Machen wir eine kurze Pause zwischen den Bildern
Sleep(10);
}
//--- Objekt CLOCK nach dem Ausschaltung des Skripts löschen
ObjectDelete(0,"CLOCK");
//---
}
```

**Sehen Sie auch**

[Ressourcen](#), [ResourceCreate\(\)](#), [ResourceSave\(\)](#), [TextGetSize\(\)](#), [TextSetFont\(\)](#)

## TextGetSize

Gibt die Breite und Höhe der Zeile mit den aktuellen [Schrifteinstellungen](#) zurück.

```
bool TextGetSize(  
    const string      text,           // Textzeile  
    uint&             width,          // Pufferbreite in Punkte  
    uint&             height,         // Pufferhöhe in Punkte  
);
```

### Optionen

*text*

[in] Die Zeile, für deren wir die Höhe und Breite erhalten.

*width*

[out] Die Eingangsparameter für die Breite.

*height*

[out] Die Eingangsparameter für die Höhe.

### Rückgabewert

Gibt bei Erfolg true zurück, ansonsten false. Mögliche Fehlercodes:

- `ERR_INTERNAL_ERROR(4001)` - in dem Fall eines Betriebssystemfehlers.

### Sehen Sie auch

[Ressourcen](#), [ResourceCreate\(\)](#), [ResourceSave\(\)](#), [TextSetFont\(\)](#), [TextOut\(\)](#)

## Funktionen für die Arbeit mit technischen Indikatoren

Alle Funktionen wie iMA, iAC, iMACD, ilchimoku usw. erzeugen im globalen cache des Kundenterminlas eine Kopie des entsprechenden Kundenindikators. Wenn eine Kopie des Indikators mit solchen Parametern schon existiert, wird eine neue Kopie nicht erzeugt, sondern Counter der Verweise an die vorgegebene Kopie steigt.

Diese Funktionen geben handle der entsprechenden Kopie des Indikators zurück. Im weiteren können Sie durch Verwendung dieses handles Daten bekommen, die vom entsprechenden Indikator berechnet werden. Daten des entsprechenden Puffers (technische Anzeiger haben berechnete Daten in ihren Innenpuffern, deren Anzahl abhängig vom Anzeiger von 1 bis 5 sein kann) können in mql5-Programm durch die Funktion [CopyBuffer\(\)](#) kopiert werden.

Man kann nicht Daten des Indikators sofort nach seiner Erzeugung verwenden, denn man braucht einige Zeit für Berechnung der Indikatorwerte, darum ist es am besten handles der Indikatoren in [OnInit\(\)](#) zu erzeugen. Funktion [iCustom\(\)](#) erzeugt entsprechenden Benutzerindikator und bei der erfolgreichen Erzeugung gibt sein handle zurück. Benutzerindikatoren können bis 512 Indikatorbuffer haben, den Inhalt durch die Funktion [CopyBuffer\(\)](#) erhalten werden kann mit der Verwendung von handle.

Es gibt eine universelle Methode, durch die Funktion durch die Funktion [IndicatorCreate\(\)](#) jeden technischen Indikator zu erzeugen. Als Eingabeparameter bekommt diese Funktion: .

- Symbolname;
- Timeframe;
- Typ des erzeugten Indikators;
- Anzahl der Eingabeparameter des Indikators;
- Feld des Typs [MqlParam](#), mit allen erforderlichen Eingabeparametern.

für die Befreiung des Speichers des Computers wird die Funktion [IndicatorRelease\(\)](#) verwendet, der handle dieses Indikators übertragen wird.

Bemerkung. Wiederholter Aufruf der Funktion des Indikators mit denselben Parametern führt nicht zu vielfachen Steigerung des Counters der Verweise, Counter wird nur einmal um 1 vergrößert. Es ist aber empfehlenswert, handles der Indikatoren in der Funktion [OnInit\(\)](#) oder im Konstruktor der Klasse zu bekommen, und dann diese handles in anderen Funktionen zu verwenden.

Alle Indikatoren haben mindestens 2 Parameter - Symbol und Periode. Wert des Symbols [NULL](#) bedeutet das laufende Instrument, Wert der Periode 0 bedeutet das laufende [Timeframe](#).

| Funktion                   | Gibt handle des Indikators zurück:         |
|----------------------------|--------------------------------------------|
| <a href="#">iAC</a>        | Accelerator Oscillator                     |
| <a href="#">iAD</a>        | Accumulation/Distribution                  |
| <a href="#">iADX</a>       | Average Directional Index                  |
| <a href="#">iADXWilder</a> | Average Directional Index by Welles Wilder |
| <a href="#">iAlligator</a> | Alligator                                  |
| <a href="#">iAMA</a>       | Adaptive Moving Average                    |
| <a href="#">iAO</a>        | Awesome Oscillator                         |

|                                    |                                               |
|------------------------------------|-----------------------------------------------|
| <a href="#"><u>iATR</u></a>        | Average True Range                            |
| <a href="#"><u>iBearsPower</u></a> | Bears Power                                   |
| <a href="#"><u>iBands</u></a>      | Bollinger Bands®                              |
| <a href="#"><u>iBullsPower</u></a> | Bulls Power                                   |
| <a href="#"><u>iCCI</u></a>        | Commodity Channel Index                       |
| <a href="#"><u>iChaikin</u></a>    | Chaikin Oscillator                            |
| <a href="#"><u>iCustom</u></a>     | Benutzerindikator                             |
| <a href="#"><u>iDEMA</u></a>       | Double Exponential Moving Average             |
| <a href="#"><u>iDeMarker</u></a>   | DeMarker                                      |
| <a href="#"><u>iEnvelopes</u></a>  | Envelopes                                     |
| <a href="#"><u>iForce</u></a>      | Force Index                                   |
| <a href="#"><u>iFractals</u></a>   | Fractals                                      |
| <a href="#"><u>iFrAMA</u></a>      | Fractal Adaptive Moving Average               |
| <a href="#"><u>iGator</u></a>      | Gator Oscillator                              |
| <a href="#"><u>ilchimoku</u></a>   | Ichimoku Kinko Hyo                            |
| <a href="#"><u>iBWMFI</u></a>      | Market Facilitation Index by Bill Williams    |
| <a href="#"><u>iMomentum</u></a>   | Momentum                                      |
| <a href="#"><u>iMFI</u></a>        | Money Flow Index                              |
| <a href="#"><u>iMA</u></a>         | Moving Average                                |
| <a href="#"><u>iOsMA</u></a>       | Moving Average of Oscillator (MACD histogram) |
| <a href="#"><u>iMACD</u></a>       | Moving Averages Convergence-Divergence        |
| <a href="#"><u>iOBV</u></a>        | On Balance Volume                             |
| <a href="#"><u>iSAR</u></a>        | Parabolic Stop And Reverse System             |
| <a href="#"><u>iRSI</u></a>        | Relative Strength Index                       |
| <a href="#"><u>iRVI</u></a>        | Relative Vigor Index                          |
| <a href="#"><u>iStdDev</u></a>     | Standard Deviation                            |
| <a href="#"><u>iStochastic</u></a> | Stochastic Oscillator                         |
| <a href="#"><u>iTEMA</u></a>       | Triple Exponential Moving Average             |
| <a href="#"><u>iTriX</u></a>       | Triple Exponential Moving Averages Oscillator |
| <a href="#"><u>iWPR</u></a>        | Williams' Percent Range                       |
| <a href="#"><u>iVIDyA</u></a>      | Variable Index DYnamic Average                |
| <a href="#"><u>iVolumes</u></a>    | Volumes                                       |



## iAC

Erzeugt im globalen cache des Kundenterminals den Indikator Accelerator Oscillator und gibt sein Handle zurück. Hat nur einen Puffer.

```
int iAC(
    string      symbol,      // Name des Symbols
    ENUM_TIMEFRAMES period   // Periode
);
```

### Parameter

*symbol*

[in] Symbolname des Instrumentes, auf dessen Daten Indikator berechnet wird. [NULL](#) bedeutet den laufenden Chart.

*period*

[in] Wert der Periode kann einer der Aufzählungswerte [ENUM\\_TIMEFRAMES](#) sein, 0 bedeutet das laufende Timeframe.

### Rückgabewert

Gibt das Handle des angegebenen technischen Indikators zurück, beim Misserfolg gibt [INVALID\\_HANDLE](#) zurück. Um Speicher des Computers vom nicht mehr verwendeten Indikator zu befreien, gibt es die Funktion [IndicatorRelease\(\)](#), der Handle dieses Indikators übertragen wird.

### Beispiel:

```
//+-----+
//|                                     Demo_iAC.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Der Indikator zeigt, wie Daten erhalten werden sollen"
#property description "Indikator-Puffer für den technischen Indikator iAC."
#property description "Symbol und Zeitrahmen, in denen der Indikator berechnet wird,"
#property description "sind durch die Parameter symbol und period angegeben."
#property description "Erstellungsweise des Handles ist durch den 'type' Parameter (T

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots 1
//--- iAC bauen
#property indicator_label1 "iAC"
#property indicator_type1  DRAW_COLOR_HISTOGRAM
#property indicator_color1 clrGreen, clrRed
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
```

```

//+-----+
//| Aufzählung der Erstellungsweisen des Handles |
//+-----+
enum Creation
{
    Call_iAC,           // iAC verwenden
    Call_IndicatorCreate // IndicatorCreate verwenden
};
//--- Eingabeparameter
input Creation      type=Call_iAC;           // Funktionstyp
input string        symbol=" ";              // Symbol
input ENUM_TIMEFRAMES period=PERIOD_CURRENT; // Zeitrahmen
//--- Indikator-Puffer
double iACBuffer[];
double iACColors[];
//--- Eine Variable um Handle des Indikators iAC zu speichern
int    handle;
//--- Variable für Speicherung
string name=symbol;
//--- Name des Indikators auf dem Chart
string short_name;
//--- die Anzahl der Werte im Indikator Accelerator Oscillator gespeichert wird
int    bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- Bindung von Arrays zu den Indikator-Puffern
    SetIndexBuffer(0,iACBuffer,INDICATOR_DATA);
    SetIndexBuffer(1,iACColors,INDICATOR_COLOR_INDEX);
    //--- das Symbol, auf das der Indikator gebaut wird, definieren
    name=symbol;
    //--- Leerzeichen aus dem linken und rechten löschen
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- Wenn nach diesem die Länge des String name ist Null
    if(StringLen(name)==0)
    {
        //--- das Symbol aus dem Chart, auf dem der Indikator läuft, nehmen
        name=_Symbol;
    };
    //--- Erstellen ein Handel des Indikators
    if(type==Call_iAC)
        handle=iAC(name,period);
    else
        handle=IndicatorCreate(name,period,IND_AC);
    //--- Wenn Handle konnte nicht erstellt werden
    if(handle==INVALID_HANDLE)

```

```

{
    //--- darüber schreiben und Nummer des Fehlers anzeigen
    PrintFormat("Ein Handle des Indikators iAC für das Paar %s/%s konnte nicht erst
        name,
        EnumToString(period),
        GetLastError());

    //--- Arbeit des Indikators ist früher geendet
    return(INIT_FAILED);
};

//--- Das Paar, Symbol/Zeitraumen, auf deren Accelerator Oscillator berechnet war, ze
short_name=StringFormat("iAC(%s/%s)",name,EnumToString(period));
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- normale Initialisierung des Indikators
return(INIT_SUCCEEDED);
};

//+-----+
//| Custom indicator iteration function |
//+-----+

int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])

{
    //--- Anzahl der Werte des Indikators iAC zu kopieren
    int values_to_copy;
    //--- Anzahl der berechneten Werte im Indikator finden
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() hat %d zurückgegeben, Fehlercode ist %d",calculate
        return(0);
    }
    //--- wenn dies der erste Start der Berechnung des Indikators ist oder die Anzahl der
    //--- oder wenn Sie brauchen den Indikator für zwei oder mehr Balken (was bedeutet, da
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
    {
        //--- wenn Array iACBuffer größer als die Anzahl der Werte in iAC auf symbol/per
        //--- sonst kopieren wir weniger als Größe der Indikator-Puffer
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    }
    else
    {

```

```

    //--- dann ist unser Indikator nicht das erste Mal berechnet, und seit dem letzten
    //--- nicht mehr als ein Balken für die Berechnung hinzugefügt war
    values_to_copy=(rates_total-prev_calculated)+1;
    ;}

//--- Arrays iACBuffer und iACColors mit den Werten aus dem Indikator Accelerator Oscillator
//--- Wenn FillArraysFromBuffer false zurückgegeben hat, dann die Daten nicht berechnen
    if(!FillArraysFromBuffer(iACBuffer,iACColors,handle,values_to_copy)) return(0);
//--- Nachricht bilden
    string comm=StringFormat("%s ==> Aktualisierte Werte im Indikator %s: %d",
                              TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                              short_name,
                              values_to_copy);
//--- Hilfsnachrichtung auf dem Chart anzeigen
    Comment(comm);
//--- die Anzahl der Werte im Indikator Accelerator Oscillator speichern
    bars_calculated=calculated;
//--- den Wert prev_calculated für den nächsten Anruf zurückgeben
    return(rates_total);
    ;}

//+-----+
//| Indikator-Puffer aus dem Indikator iAC ausfüllen |
//+-----+
bool FillArraysFromBuffer(double &values[], // Indikator-Puffer mit Werten von
                          double &color_indexes[], // Farbpuffer (um Farbindex zu speichern)
                          int ind_handle, // Handle des Indikators iAC
                          int amount // Anzahl der Werte, die kopiert werden
                          )
{
    //--- Fehlercode rücksetzen
    ResetLastError();
//--- Teil des Arrays iACBuffer mit Werten auf Indikator-Puffer mit Index 0 ausfüllen
    if(CopyBuffer(ind_handle,0,0,amount,values)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iAC kopiert werden, Fehlercode: %d", GetLastError());
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden konnte
        return(false);
    }
    ;}

//--- Nun die Farbeindizes kopieren
    if(CopyBuffer(ind_handle,1,0,amount,color_indexes)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Die Farbwerte konnte nicht aus dem Indikator iAC kopiert werden, Fehlercode: %d", GetLastError());
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden konnte
        return(false);
    }
    ;}

//--- Alles gelang
    return(true);
    ;}

```

```
//+-----+  
//| Indicator deinitialization function |  
//+-----+  
void OnDeinit(const int reason)  
{  
//--- Das Chart nach der Löschung des Indikators leeren  
    Comment("");  
};
```

## iAD

Gibt das Handle des Indikators Accumulation/Distribution zurück. Hat nur einen Puffer.

```
int iAD(
    string          symbol,          // Name des Symbols
    ENUM_TIMEFRAMES period,          // Periode
    ENUM_APPLIED_VOLUME applied_volume // Volumentyp für Berechnung
);
```

### Parameter

*symbol*

[in] Symbolname des Instruments, dessen Daten für Berechnung des Indikators verwendet werden. NULL bedeutet das laufende Symbol.

*period*

[in] Wert der Periode kann einer der Aufzählungswerte ENUM\_TIMEFRAMES sein, 0 bedeutet das laufende Timeframe.

*applied\_volume*

[in] Das verwendete Volumen. Kann einer der Werte ENUM\_APPLIED\_VOLUME sein.

### Rückgabewert

Gibt das Handle des angegebenen technischen Indikators zurück, beim Misserfolg gibt INVALID\_HANDLE zurück. Um Computerspeicher aus nicht genutzten Indikatoren zu befreien, wird die Funktion IndicatorRelease() verwendet, zu deren Indikatorhandle passiert wird.

### Beispiel:

```
//+-----+
//|                                     Demo_iAD.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Der Indikator zeigt, wie Daten erhalten werden sollen"
#property description "Indikator-Puffer für den technischen Indikator iAD."
#property description "Symbol und Zeitrahmen, in denen der Indikator berechnet wird,"
#property description "sind durch die Parameter symbol und period angegeben."
#property description "Erstellungsweise des Handles ist durch den 'type' Parameter (T)"

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots 1
//--- plot iAD
#property indicator_label1 "iAD"
#property indicator_type1  DRAW_LINE
#property indicator_color1 clrLightSeaGreen
```

```

#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+-----+
//| Aufzählung der Erstellungsweisen des Handles |
//+-----+
enum Creation
{
    Call_iAD,          // iAD verwenden
    Call_IndicatorCreate // IndicatorCreate verwenden
};
//--- Eingabeparameter
input Creation      type=Call_iAD;          // Funktionstyp
input ENUM_APPLIED_VOLUME volumes;          // verwendetes Volumen
input string        symbol=" ";             // Symbol
input ENUM_TIMEFRAMES period=PERIOD_CURRENT; // Zeitrahmen
//--- Indicator-Puffer
double      iADBuffer[];
//--- Eine Variable um Handle des Indikators iAD zu speichern
int         handle;
//--- Variable für Speicherung
string name=symbol;
//--- Name des Indikators auf dem Chart
string short_name;
//--- die Anzahl der Werte im Indikator Accumulation/Distribution gespeichert wird
int         bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- Bindung von Array zum Indikator-Puffer
    SetIndexBuffer(0,iADBuffer,INDICATOR_DATA);
    //--- das Symbol, auf das der Indikator gebaut wird, definieren
    name=symbol;
    //--- Leerzeichen aus dem linken und rechten löschen
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- Wenn nach diesem die Länge des String name ist Null
    if(StringLen(name)==0)
    {
        //--- das Symbol aus dem Chart, auf dem der Indikator läuft, nehmen
        name=_Symbol;
    }
    //--- Erstellen ein Handel des Indikators
    if(type==Call_iAD)
        handle=iAD(name,period,volumes);
    else
    {
        //--- die Struktur mit Werten von Indikatorparametern ausfüllen

```

```

    MqlParam pars[1];
    pars[0].type=TYPE_INT;
    pars[0].integer_value=volumes;
    handle=IndicatorCreate(name,period,IND_AD,1,pars);
};

//--- Wenn Handle konnte nicht erstellt werden
if(handle==INVALID_HANDLE)
{
    //--- darüber schreiben und Nummer des Fehlers anzeigen
    PrintFormat("Handle des Indikators iAD für das Paar %s/%s konnte nicht erstellt
                name,
                EnumToString(period),
                GetLastError());
    //--- Arbeit des Indikators ist früher geendet
    return(INIT_FAILED);
};

//--- Das Paar von Symbol/Zeitraumen, auf deren Accumulation/Distribution berechnet wa
short_name=StringFormat("iAD(%s/%s)",name,EnumToString(period));
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- normale Initialisierung des Indikators
return(INIT_SUCCEEDED);
};

//+-----+
//| Custom indicator iteration function |
//+-----+

int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{
    //--- Anzahl der Werte des Indikators iAD zu kopieren
    int values_to_copy;
    //--- Anzahl der berechneten Werte im Indikator finden
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() hat %d zurückgegeben, Fehlercode ist %d",calculated,
                    GetLastError());
        return(0);
    };
    //--- wenn dies der erste Start der Berechnung des Indikators ist oder die Anzahl der
    //--- oder wenn Sie brauchen den Indikator für zwei oder mehr Balken (was bedeutet, da
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
    {

```



```

    //--- wenn Array iADBuffer größer als die Anzahl der Werte in iAD auf symbol/pe
    //--- sonst kopieren wir weniger als Größe der Indikator-Puffer
    if(calculated>rates_total) values_to_copy=rates_total;
    else
        values_to_copy=calculated;
    ;}
else
{
    //--- dann ist unser Indikator nicht das erste Mal berechnet, und seit dem letzte
    //--- nicht mehr als ein Balken für die Berechnung hinzugefügt war
    values_to_copy=(rates_total-prev_calculated)+1;
    ;}

//--- Das Array iADBuffer mit den Werten aus dem Indikator Accumulation/Distribution
//--- Wenn FillArraysFromBuffer false zurückgegeben hat, dann die Daten nicht bereit
    if(!FillArrayFromBuffer(iADBuffer,handle,values_to_copy)) return(0);
//--- Nachricht bilden
    string comm=StringFormat("%s ==> Aktualisierte Werte im Indikator %s: %d",
                                TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                                short_name,
                                values_to_copy);
//--- Hilfsnachrichtung auf dem Chart anzeigen
    Comment(comm);
//--- die Anzahl der Werte im Indikator Accumulation/Distribution speichern
    bars_calculated=calculated;
//--- den Wert prev_calculated für den nächsten Anruf zurückgeben
    return(rates_total);
    ;}

//+-----+
//| Indikator-Puffer aus dem Indikator iAD ausfüllen |
//+-----+
bool FillArrayFromBuffer(double &values[], // Indikator-Puffer der Linie von Accumu
                        int ind_handle, // Handle des Indikators iAD
                        int amount // Anzahl der Werte, die kopiert werden
                        )
{
    //--- Fehlercode rücksetzen
    ResetLastError();
//--- Teil des Arrays iADBuffer mit Werten auf Indikator-Puffer mit Index 0 ausfüllen
    if(CopyBuffer(ind_handle,0,0,amount,values)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iAD kopiert werden, Fehlercode
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berech
        return(false);
    }
    ;}
//--- Alles gelang
    return(true);
    ;}

//+-----+
//| Indicator deinitialization function |

```

```
//+-----+  
void OnDeinit(const int reason)  
{  
    //--- Das Chart nach der Löschung des Indikators leeren  
    Comment("");  
};
```

## iADX

Gibt das Handle des Indikators Average Directional Movement Index zurück.

```
int iADX(
    string          symbol,          // Name des Symbols
    ENUM_TIMEFRAMES period,          // Periode
    int             adx_period       // Glättungsperiode
);
```

### Parameter

*symbol*

[in] Symbolname des Instrumentes, auf dessen Daten Indikator berechnet wird. [NULL](#) bedeutet das laufende Symbol.

*period*

[in] Wert der Periode kann einer der Aufzählungswerte [ENUM\\_TIMEFRAMES](#) sein, 0 bedeutet das laufende Timeframe.

*adx\_period*

[in] Periode für die Berechnung des Indexes.

### Bemerkung

Nummer der Puffer: 0 - MAIN\_LINE, 1 - PLUSDI\_LINE, 2 - MINUSDI\_LINE.

### Rückgabewert

Gibt das Handle des angegebenen technischen Indikators zurück, beim Misserfolg gibt [INVALID\\_HANDLE](#) zurück. Um Computerspeicher aus nicht genutzten Indikatoren zu befreien, wird die Funktion [IndicatorRelease\(\)](#) verwendet, zu deren Indikatorhandle passiert wird.

### Beispiel:

```
//+-----+
//|                                     Demo_iADX.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Der Indikator zeigt, wie Daten erhalten werden sollen"
#property description "Indikator-Puffer für den technischen Indikator iADX."
#property description "Symbol und Zeitrahmen, in denen der Indikator berechnet wird,"
#property description "sind durch die Parameter symbol und period angegeben."
#property description "Erstellungsweise des Handles ist durch den 'type' Parameter (T)"

#property indicator_separate_window
#property indicator_buffers 3
#property indicator_plots 3
```

```

//--- plot ADX
#property indicator_label1  "ADX"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- plot DI_plus
#property indicator_label2  "DI_plus"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrYellowGreen
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- plot DI_minus
#property indicator_label3  "DI_minus"
#property indicator_type3   DRAW_LINE
#property indicator_color3  clrWheat
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//+-----+
//| Aufzählung der Erstellungsweisen des Handles |
//+-----+
enum Creation
{
    Call_iADX,          // iADX verwenden
    Call_IndicatorCreate // IndicatorCreate verwenden
};
//--- Eingabeparameter
input Creation      type=Call_iADX;          // Typ der Funktion
input int           adx_period=14;           // Berechnungszeitraum
input string        symbol=" ";              // Symbol
input ENUM_TIMEFRAMES period=PERIOD_CURRENT; // Zeitrahmen
//--- Indikator-Puffer
double            ADXBuffer[];
double            DI_plusBuffer[];
double            DI_minusBuffer[];
//--- Eine Variable um Handle des Indikators iADX zu speichern
int    handle;
//--- Variable für Speicherung
string name=symbol;
//--- Name des Indikators auf dem Chart
string short_name;
//--- die Anzahl der Werte im Indikator Average Directional Movement Index gespeichert
int    bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- Bindung von Array zu den Indikator-Puffern

```

```

SetIndexBuffer(0,ADXBuffer,INDICATOR_DATA);
SetIndexBuffer(1,DI_plusBuffer,INDICATOR_DATA);
SetIndexBuffer(2,DI_minusBuffer,INDICATOR_DATA);
//--- das Symbol, auf das der Indikator gebaut wird, definieren
name=symbol;
//--- Leerzeichen aus dem linken und rechten löschen
StringTrimRight(name);
StringTrimLeft(name);
//--- Wenn nach diesem die Länge des String name ist Null
if(StringLen(name)==0)
{
    //--- das Symbol aus dem Chart, auf dem der Indikator läuft, nehmen
    name=_Symbol;
};
//--- Erstellen ein Handel des Indikators
if(type==Call_iADX)
    handle=iADX(name,period,adx_period);
else
{
    //--- die Struktur mit Werten von Indikatorparametern ausfüllen
    MqlParam pars[1];
    pars[0].type=TYPE_INT;
    pars[0].integer_value=adx_period;
    handle=IndicatorCreate(name,period,IND_ADX,1,pars);
};
//--- Wenn Handle konnte nicht erstellt werden
if(handle==INVALID_HANDLE)
{
    //--- darüber schreiben und Nummer des Fehlers anzeigen
    PrintFormat("Handle des Indikators iADX für das Paar %s/%s konnte nicht erstellt
                name,
                EnumToString(period),
                GetLastError());
    //--- Arbeit des Indikators ist früher geendet
    return(INIT_FAILED);
};
//--- Das Paar, Symbol/Zeitraumen, auf deren Average Directional Movement Index berech
short_name=StringFormat("iADX(%s/%s period=%d)",name,EnumToString(period),adx_peric
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- normale Initialisierung des Indikators
return(INIT_SUCCEEDED);
};
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],

```

```

        const double &high[],
        const double &low[],
        const double &close[],
        const long &tick_volume[],
        const long &volume[],
        const int &spread[])
    {
//--- Anzahl der Werte des Indikators iADX zu kopieren
        int values_to_copy;
//--- Anzahl der berechneten Werte im Indikator finden
        int calculated=BarsCalculated(handle);
        if(calculated<=0)
        {
            PrintFormat("BarsCalculated() hat %d zurückgegeben, Fehlercode ist %d",calculated,GetLastError());
            return(0);
        }
//--- wenn dies der erste Start der Berechnung des Indikators ist oder die Anzahl der
//--- oder wenn Sie brauchen den Indikator für zwei oder mehr Balken (was bedeutet, dass
        if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
        {
            //--- wenn Array ADXBuffer größer als die Anzahl der Werte in iADX auf symbol/period
            //--- sonst kopieren wir weniger als Größe der Indikator-Puffer
            if(calculated>rates_total) values_to_copy=rates_total;
            else
                values_to_copy=calculated;
        }
        else
        {
            //--- dann ist unser Indikator nicht das erste Mal berechnet, und seit dem letzten Mal
            //--- nicht mehr als ein Balken für die Berechnung hinzugefügt war
            values_to_copy=(rates_total-prev_calculated)+1;
        }
//--- Das Array mit den Werten aus dem Indikator Average Directional Movement Index auf
//--- Wenn FillArraysFromBuffer false zurückgegeben hat, dann die Daten nicht bereit sind
        if(!FillArraysFromBuffers(ADXBuffer,DI_plusBuffer,DI_minusBuffer,handle,values_to_copy))
//--- Nachricht bilden
        string comm=StringFormat("%s ==> Aktualisierte Werte im Indikator %s: %d",
                                   TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                                   short_name,
                                   values_to_copy);
//--- Hilfsnachrichtung auf dem Chart anzeigen
        Comment(comm);
//--- die Anzahl der Werte im Indikator Average Directional Movement Index speichern
        bars_calculated=calculated;
//--- den Wert prev_calculated für den nächsten Anruf zurückgeben
        return(rates_total);
    }
//+-----+
//| Indikator-Puffer aus dem Indikator iADX ausfüllen |
//+-----+

```

```

bool FillArraysFromBuffers(double &adx_values[],      // Indikator-Puffer der Linie von
                        double &DIplus_values[],     // Indikator-Puffer für DI+
                        double &DIminus_values[],    // Indikator-Puffer für DI-
                        int ind_handle,              // Handle des Indikators iADXWilde
                        int amount                   // Anzahl der Werte, die kopiert werden
                        )
{
    //--- Fehlercode zurücksetzen
    ResetLastError();
    //--- Teil des Arrays ADXBuffer mit Werten auf Indikator-Puffer mit Index 0 ausfüllen
    if(CopyBuffer(ind_handle,0,0,amount,adx_values)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iADX kopiert werden, Fehlercode: %d", GetLastError());
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden konnte
        return(false);
    }
    ;}

    //--- Teil des Arrays DI_plusBuffer mit Werten auf Indikator-Puffer mit Index 1 ausfüllen
    if(CopyBuffer(ind_handle,1,0,amount,DIplus_values)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iADX kopiert werden, Fehlercode: %d", GetLastError());
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden konnte
        return(false);
    }
    ;}

    //--- Teil des Arrays DI_minusBuffer mit Werten auf Indikator-Puffer mit Index 2 ausfüllen
    if(CopyBuffer(ind_handle,2,0,amount,DIminus_values)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iADX kopiert werden, Fehlercode: %d", GetLastError());
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden konnte
        return(false);
    }
    ;}

    //--- Alles gelang
    return(true);
}
;

//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
    //--- Das Chart nach der Löschung des Indikators leeren
    Comment("");
}
;

```

## iADXWilder

Gibt das Handle des Indikators Average Directional Movement Index by Welles Wilder zurück.

```
int iADXWilder(
    string      symbol,          // Symbolname
    ENUM_TIMEFRAMES period,      // Periode
    int         adx_period       // Glättungsperiode
);
```

### Parameter

*symbol*

[in] Symbolname des Instruments, dessen Daten für Berechnung des Indikators verwendet werden. NULL bedeutet das laufende Symbol.

*period*

[in] Wert der Periode kann einer der Aufzählungswerte ENUM\_TIMEFRAMES sein, 0 bedeutet das laufende Timeframe.

*adx\_period*

[in] Periode für Berechnung des Indexes.

### Bemerkung

Puffernummern: 0 - MAIN\_LINE, 1 - PLUSDI\_LINE, 2 - MINUSDI\_LINE.

### Rückgabewert

Gibt das Handle des angegebenen technischen Indikators zurück, beim Misserfolg gibt INVALID\_HANDLE zurück. Um Computerspeicher aus nicht genutzten Indikatoren zu befreien, wird die Funktion IndicatorRelease() verwendet, zu deren Indikatorhandle passiert wird.

### Beispiel:

```
//+-----+
//|                                     iADXWilder.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Der Indikator zeigt, wie Daten erhalten werden sollen"
#property description "Indikator-Puffer für den technischen Indikator iADXWilder."
#property description "Symbol und Zeitrahmen, in denen der Indikator berechnet wird,"
#property description "sind durch die Parameter symbol und period angegeben."
#property description "Erstellungsweise des Handles ist durch den 'type' Parameter (T)"

#property indicator_separate_window
#property indicator_buffers 3
#property indicator_plots 3
```



```

//--- plot ADX
#property indicator_label1  "ADX"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- plot DI_plus
#property indicator_label2  "DI_plus"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrYellowGreen
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- plot DI_minus
#property indicator_label3  "DI_minus"
#property indicator_type3   DRAW_LINE
#property indicator_color3  clrWheat
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//+-----+
//| Aufzählung der Erstellungsweisen des Handles |
//+-----+
enum Creation
{
    Call_iADXWilder,      // iADXWilder verwenden
    Call_IndicatorCreate  // IndicatorCreate verwenden
};
//--- Eingabeparameter
input Creation      type=Call_iADXWilder;  // Funktionstyp
input int           adx_period=14;         // Berechnungszeitraum
input string        symbol=" ";            // Symbol
input ENUM_TIMEFRAMES period=PERIOD_CURRENT; // Zeitrahmen
//--- Indicator-Puffer
double      ADXBuffer[];
double      DI_plusBuffer[];
double      DI_minusBuffer[];
//--- Eine Variable um Handle des Indikators iADXWilder zu speichern
int  handle;
//--- Variable für Speicherung
string name=symbol;
//--- Name des Indikators auf dem Chart
string short_name;
//--- die Anzahl der Werte im Indikator Average Directional Movement Index by Welles V
int  bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- Bindung von Array zu den Indikator-Puffern

```

```

SetIndexBuffer(0,ADXBuffer,INDICATOR_DATA);
SetIndexBuffer(1,DI_plusBuffer,INDICATOR_DATA);
SetIndexBuffer(2,DI_minusBuffer,INDICATOR_DATA);
//--- das Symbol, auf das der Indikator gebaut wird, definieren
name=symbol;
//--- Leerzeichen aus dem linken und rechten löschen
StringTrimRight(name);
StringTrimLeft(name);
//--- Wenn nach diesem die Länge des String name ist Null
if(StringLen(name)==0)
{
    //--- das Symbol aus dem Chart, auf dem der Indikator läuft, nehmen
    name=_Symbol;
};
//--- Erstellen ein Handel des Indikators
if(type==Call_iADXWilder)
    handle=iADXWilder(name,period,adx_period);
else
{
    //--- die Struktur mit Werten von Indikatorparametern ausfüllen
    MqlParam pars[1];
    pars[0].type=TYPE_INT;
    pars[0].integer_value=adx_period;
    handle=IndicatorCreate(name,period,IND_ADXW,1,pars);
};
//--- Wenn Handle konnte nicht erstellt werden
if(handle==INVALID_HANDLE)
{
    //--- darüber schreiben und Nummer des Fehlers anzeigen
    PrintFormat("Handle des Indikators iADXWilder für das Paar %s/%s konnte nicht e
        name,
        EnumToString(period),
        GetLastError());
    //--- Arbeit des Indikators ist früher geendet
    return(INIT_FAILED);
};
//--- Das Paar, Symbol/Zeitraumen, auf deren Average Directional Movement Index by Wel
short_name=StringFormat("iADXWilder(%s/%s period=%d)",name,EnumToString(period),adx
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- normale Initialisierung des Indikators
return(INIT_SUCCEEDED);
};
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],

```

```

        const double &high[],
        const double &low[],
        const double &close[],
        const long &tick_volume[],
        const long &volume[],
        const int &spread[])

{
//--- Anzahl der Werte des Indikators iADXWilder zu kopieren
    int values_to_copy;
//--- Anzahl der berechneten Werte im Indikator finden
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() hat %d zurückgegeben, Fehlercode ist %d",calculated,GetLastError());
        return(0);
    }
//--- wenn dies der erste Start der Berechnung des Indikators ist oder die Anzahl der
//--- oder wenn Sie brauchen den Indikator für zwei oder mehr Balken (was bedeutet, dass
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
    {
        //--- wenn Array ADXBuffer größer als die Anzahl der Werte in iADXWilder auf synthetischen
        //--- sonst kopieren wir weniger als Größe der Indikator-Puffer
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    }
    else
    {
        //--- dann ist unser Indikator nicht das erste Mal berechnet, und seit dem letzten Mal
        //--- nicht mehr als ein Balken für die Berechnung hinzugefügt war
        values_to_copy=(rates_total-prev_calculated)+1;
    }
//--- Das Array mit den Werten aus dem Indikator Average Directional Movement Index by
//--- Wenn FillArraysFromBuffer false zurückgegeben hat, dann die Daten nicht bereit sind
    if(!FillArraysFromBuffers(ADXBuffer,DI_plusBuffer,DI_minusBuffer,handle,values_to_copy))
//--- Nachricht bilden
    string comm=StringFormat("%s ==> Aktualisierte Werte im Indikator %s: %d",
                                TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                                short_name,
                                values_to_copy);
//--- Hilfsnachrichtung auf dem Chart anzeigen
    Comment(comm);
//--- die Anzahl der Werte im Indikator Average Directional Movement Index speichern
    bars_calculated=calculated;
//--- den Wert prev_calculated für den nächsten Anruf zurückgeben
    return(rates_total);
}
//+-----+
//| Indikator-Puffer aus dem Indikator iADXWilder ausfüllen |
//+-----+

```

```

bool FillArraysFromBuffers(double &adx_values[],      // Indikator-Puffer der Linie von
                        double &DIplus_values[],      // Indikator-Puffer für DI+
                        double &DIminus_values[],      // Indikator-Puffer für DI-
                        int ind_handle,                // Handle des Indikators iADXWilder
                        int amount                     // Anzahl der Werte, die kopiert werden
                        )
{
    //--- Fehlercode rücksetzen
    ResetLastError();
    //--- Teil des Arrays ADXBuffer mit Werten auf Indikator-Puffer mit Index 0 ausfüllen
    if(CopyBuffer(ind_handle,0,0,amount,adx_values)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iADXWilder kopiert werden, Fehlercode: %d", GetLastError());
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden konnte
        return(false);
    }
; }

//--- Teil des Arrays DI_plusBuffer mit Werten auf Indikator-Puffer mit Index 1 ausfüllen
if(CopyBuffer(ind_handle,1,0,amount,DIplus_values)<0)
{
    //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
    PrintFormat("Daten konnte nicht aus dem Indikator iADXWilder kopiert werden, Fehlercode: %d", GetLastError());
    //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden konnte
    return(false);
}
; }

//--- Teil des Arrays DI_minusBuffer mit Werten auf Indikator-Puffer mit Index 2 ausfüllen
if(CopyBuffer(ind_handle,2,0,amount,DIminus_values)<0)
{
    //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
    PrintFormat("Daten konnte nicht aus dem Indikator iADXWilder kopiert werden, Fehlercode: %d", GetLastError());
    //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden konnte
    return(false);
}
; }

//--- Alles gelang
return(true);
; }

//+-----+
//| Indicator deinitialization function |
//+-----+

void OnDeinit(const int reason)
{
    //--- Das Chart nach der Löschung des Indikators leeren
    Comment("");
; }

```

## iAlligator

Gibt das Handle des Indikators Alligator zurück.

```
int iAlligator(
    string          symbol,          // Symbolname
    ENUM_TIMEFRAMES period,          // Periode
    int             jaw_period,      // Periode für Berechnung der Kiefer
    int             jaw_shift,       // horizontale Verschiebung der Kiefer
    int             teeth_period,    // Periode für die Berechnung der Zähne
    int             teeth_shift,     // horizontale Verschiebung der Zähne
    int             lips_period,     // Periode für Berechnung der Lippen
    int             lips_shift,      // horizontale Verschiebung der Lippen
    ENUM_MA_METHOD  ma_method,      // Glaettungstyp
    ENUM_APPLIED_PRICE applied_price // Preistyp oder Handle
);
```

### Parameter

*symbol*

[in] Symbolname des Instrumentes, dessen Daten für Berechnung des Indikators verwendet werden. [NULL](#) bedeutet das laufende Symbol.

*period*

[in] Wert der Periode kann einer der Aufzählungswerte [ENUM\\_TIMEFRAMES](#) sein, 0 bedeutet das laufende Timeframe.

*jaw\_period*

[in] Mittelungsperiode der blauen Linie (Kiefer des Alligators).

*jaw\_shift*

[in] Verschiebung der blauen Linie in Bezug auf das Preischart.

*teeth\_period*

[in] Mittelungsperiode der roten Linie (Zähne des Alligators).

*teeth\_shift*

[in] Verschiebung der roten Linie in Bezug auf das Preischart.

*lips\_period*

[in] Mittelungsperiode der grünen Linie (Lippen des Alligators).

*lips\_shift*

[in] Verschiebung der grünen Linie in Bezug auf das Preischart.

*ma\_method*

[in] Mittelungsmethode. Kann einer der Aufzählungswerte [ENUM\\_MA\\_METHOD](#) sein.

*applied\_price*

[in] Der verwendete Preis. Kann einer der Preiskonstanten [ENUM\\_APPLIED\\_PRICE](#) oder Handle eines anderen Indikators sein

## Rückgabewert

Gibt das Handle des angegebenen technischen Indikators zurück, beim Misserfolg gibt [INVALID\\_HANDLE](#) zurück. Um Computerspeicher aus nicht genutzten Indikatoren zu befreien, wird die Funktion [IndicatorRelease\(\)](#) verwendet, zu deren Indikatorhandle passiert wird.

## Bemerkung

Puffernummern: 0 - GATORJAW\_LINE, 1 - GATORTEETH\_LINE, 2 - GATORLIPS\_LINE.

## Beispiel:

```
//+-----+
//|                                     Demo_iAlligator.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Der Indikator zeigt, wie Daten erhalten werden sollen"
#property description "Indikator-Puffer für den technischen Indikator iAlligator."
#property description "Symbol und Zeitrahmen, in denen der Indikator berechnet wird,"
#property description "sind durch die Parameter symbol und period angegeben."
#property description "Erstellungsweise des Handles ist durch den 'type' Parameter (T)"
#property description "Alle andere Parameter sind wie im normalen Indikator Alligator."

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots 3
//--- plot Jaws
#property indicator_label1 "Jaws"
#property indicator_type1  DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- plot Teeth
#property indicator_label2 "Teeth"
#property indicator_type2  DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- plot Lips
#property indicator_label3 "Lips"
#property indicator_type3  DRAW_LINE
#property indicator_color3  clrLime
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//+-----+
//| Aufzählung der Erstellungsweisen des Handles |
```

```

//+-----+
enum Creation
{
    Call_iAlligator,      // iAlligator verwenden
    Call_IndicatorCreate  // IndicatorCreate verwenden
};
//--- Eingabeparameter
input Creation          type=Call_iAlligator;  // Funktionstyp
input string            symbol=" ";            // Symbol
input ENUM_TIMEFRAMES  period=PERIOD_CURRENT; // Zeitrahmen
input int               jaw_period=13;         // Zeitraum für die Kiefer-Linie
input int               jaw_shift=8;           // Verschiebung der Kiefer-Linie
input int               teeth_period=8;        // Zeitraum für die Zähne-Linie
input int               teeth_shift=5;         // Verschiebung der Zähne-Linie
input int               lips_period=5;         // Zeitraum für die Lippen-Linie
input int               lips_shift=3;          // Verschiebung der Lippen-Linie
input ENUM_MA_METHOD    MA_method=MODE_SMMMA; // Methode der Mittelung der Alligator
input ENUM_APPLIED_PRICE applied_price=PRICE_MEDIAN; // Typ der Preise aus der der Alligator
//--- Indikator-Puffer
double      JawsBuffer[];
double      TeethBuffer[];
double      LipsBuffer[];
//--- Eine Variable um Handle des Indikators iAlligator zu speichern
int         handle;
//--- Variable für Speicherung
string name=symbol;
//--- Name des Indikators auf dem Chart
string short_name;
//--- die Anzahl der Werte im Indikator Alligator gespeichert wird
int      bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- Bindung von Array zu den Indikator-Puffern
    SetIndexBuffer(0,JawsBuffer,INDICATOR_DATA);
    SetIndexBuffer(1,TeethBuffer,INDICATOR_DATA);
    SetIndexBuffer(2,LipsBuffer,INDICATOR_DATA);
    //--- Verschiebung jede Linie definieren
    PlotIndexSetInteger(0,PLOT_SHIFT,jaw_shift);
    PlotIndexSetInteger(1,PLOT_SHIFT,teeth_shift);
    PlotIndexSetInteger(2,PLOT_SHIFT,lips_shift);
    //--- das Symbol, auf das der Indikator gebaut wird, definieren
    name=symbol;
    //--- Leerzeichen aus dem linken und rechten löschen
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- Wenn nach diesem die Länge des String name ist Null

```

```

if (StringLen (name) == 0)
{
    //--- das Symbol aus dem Chart, auf dem der Indikator läuft, nehmen
    name = _Symbol;
};
//--- Erstellen ein Handel des Indikators
if (type == Call_iAlligator)
    handle = iAlligator (name, period, jaw_period, jaw_shift, teeth_period,
                        teeth_shift, lips_period, lips_shift, MA_method, applied_price);
else
{
    //--- die Struktur mit Werten von Indikatorparametern ausfüllen
    MqlParam pars[8];
    //--- Perioden und Verschiebungen der Alligator-Linien
    pars[0].type = TYPE_INT;
    pars[0].integer_value = jaw_period;
    pars[1].type = TYPE_INT;
    pars[1].integer_value = jaw_shift;
    pars[2].type = TYPE_INT;
    pars[2].integer_value = teeth_period;
    pars[3].type = TYPE_INT;
    pars[3].integer_value = teeth_shift;
    pars[4].type = TYPE_INT;
    pars[4].integer_value = lips_period;
    pars[5].type = TYPE_INT;
    pars[5].integer_value = lips_shift;
    //--- Art der Glättung
    pars[6].type = TYPE_INT;
    pars[6].integer_value = MA_method;
    //--- Art des Preises
    pars[7].type = TYPE_INT;
    pars[7].integer_value = applied_price;
    //--- Handle erstellen
    handle = IndicatorCreate (name, period, IND_ALLIGATOR, 8, pars);
};
//--- Wenn Handle konnte nicht erstellt werden
if (handle == INVALID_HANDLE)
{
    //--- darüber schreiben und Nummer des Fehlers anzeigen
    PrintFormat ("Handle des Indikators iAlligator für das Paar %s/%s konnte nicht e
                name,
                EnumToString (period),
                GetLastError ());
    //--- Arbeit des Indikators ist früher geendet
    return (INIT_FAILED);
};
//--- Das Paar von Symbol/Zeitraumen, auf deren Alligator berechnet war, anzeigen
short_name = StringFormat ("iAlligator (%s/%s, %d,%d,%d,%d,%d,%d)", name, EnumToString (pe
                jaw_period, jaw_shift, teeth_period, teeth_shift, lips_period, l

```



```

    IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- normale Initialisierung des Indikators
    return(INIT_SUCCEEDED);
};
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
//--- Anzahl der Werte des Indikators iAlligator zu kopieren
    int values_to_copy;
//--- Anzahl der berechneten Werte im Indikator finden
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() hat %d zurückgegeben, Fehlercode ist %d",calculated,GetLastError());
        return(0);
    }
//--- wenn dies der erste Start der Berechnung des Indikators ist oder die Anzahl der
//--- oder wenn Sie brauchen den Indikator für zwei oder mehr Balken (was bedeutet, dass
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
    {
        //--- wenn Array JawsBuffer größer als die Anzahl der Werte in iAlligator auf system
        //--- sonst kopieren wir weniger als Größe der Indikator-Puffer
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    }
else
    {
        //--- dann ist unser Indikator nicht das erste Mal berechnet, und seit dem letzten
        //--- nicht mehr als ein Balken für die Berechnung hinzugefügt war
        values_to_copy=(rates_total-prev_calculated)+1;
    }
//--- Die Arrays mit den Werten aus dem Indikator Alligator ausfüllen
//--- Wenn FillArraysFromBuffer false zurückgegeben hat, dann die Daten nicht bereit sind
    if(!FillArraysFromBuffer(JawsBuffer,jaw_shift,TeethBuffer,teeth_shift,LipsBuffer,lips_shift))
//--- Nachricht bilden
    string comm=StringFormat("%s ==> Aktualisierte Werte im Indikator %s: %d",
                             TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                             short_name,

```

```

        values_to_copy);
//--- Hilfsnachrichtung auf dem Chart anzeigen
    Comment(comm);
//--- die Anzahl der Werte im Indikator Alligator speichern
    bars_calculated=calculated;
//--- den Wert prev_calculated für den nächsten Anruf zurückgeben
    return(rates_total);
; }
//+-----+
//| Indikator-Puffer aus dem Indikator iAlligator ausfüllen |
//+-----+
bool FillArraysFromBuffers(double &jaws_buffer[], // Indikator-Puffer für die Kiefer-
                           int j_shift,          // Verschiebung der Kiefer-Linie
                           double &teeth_buffer[], // Indikator-Puffer für die Zähne-Linie
                           int t_shift,          // Verschiebung der Zähne-Linie
                           double &lips_buffer[], // Indikator-Puffer für die Lippen-Linie
                           int l_shift,          // Verschiebung der Lippen-Linie
                           int ind_handle,        // Handle von iAlligator
                           int amount            // Anzahl der Werte, die kopiert werden
                           )
{
//--- Fehlercode rücksetzen
    ResetLastError();
//--- Teil des Arrays JawsBuffer mit Werten auf Indikator-Puffer mit Index 0 ausfüllen
    if(CopyBuffer(ind_handle,0,-j_shift,amount,jaws_buffer)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iAlligator kopiert werden, Fehlercode: %d", GetLastError());
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden konnte
        return(false);
    }
; }

//--- Teil des Arrays TeethBuffer mit Werten auf Indikator-Puffer mit Index 1 ausfüllen
    if(CopyBuffer(ind_handle,1,-t_shift,amount,teeth_buffer)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iAlligator kopiert werden, Fehlercode: %d", GetLastError());
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden konnte
        return(false);
    }
; }

//--- Teil des Arrays LipsBuffer mit Werten auf Indikator-Puffer mit Index 2 ausfüllen
    if(CopyBuffer(ind_handle,2,-l_shift,amount,lips_buffer)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iAlligator kopiert werden, Fehlercode: %d", GetLastError());
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden konnte
        return(false);
    }
; }

```

```
//--- Alles gelang
    return(true);
;}

//+-----+
//| Indicator deinitialization function |
//+-----+

void OnDeinit(const int reason)
{
//--- Das Chart nach der Löschung des Indikators leeren
    Comment("");
;}
;
```

## iAMA

Gibt das Handle des Indikators Adaptive Moving Average zurück. Hat nur einen Puffer.

```
int iAMA(
    string          symbol,          // Symbolname
    ENUM_TIMEFRAMES period,          // Periode
    int             ama_period,      // Periode AMA
    int             fast_ma_period,  // Schnelle MA Periode
    int             slow_ma_period,  // Langsame MA Periode
    int             ama_shift,       // horizontale Verschiebung des Indikators
    ENUM_APPLIED_PRICE applied_price // Preistyp oder Handle
);
```

### Parameter

*symbol*

[in] Symbolname des Instrumentes, dessen Daten für Berechnung des Indikators verwendet werden. [NULL](#) bedeutet das laufende Symbol.

*period*

[in] Wert der Periode kann einer der Aufzählungswerte [ENUM\\_TIMEFRAMES](#) sein, 0 bedeutet das laufende Timeframe.

*ama\_period*

[in] Periode der Berechnung, wenn Effektivitätskoeffizient berechnet wird.

*fast\_ma\_period*

[in] Schnelle Periode für Berechnung des Mittelungskoeffizient für schnellen Markt.

*slow\_ma\_period*

[in] Langsame Periode für Berechnung des Mittelungskoeffizient beim Fehlen des Trends .

*ama\_shift*

[in] Verschiebung des Indikators in Bezug auf das Preischart.

*applied\_price*

[in] Der verwendete Preis. Kann eine der Preiskonstanten [ENUM\\_APPLIED\\_PRICE](#) oder Handle eines anderen Indikators sein.

### Rückgabewert

Gibt das Handle des angegebenen technischen Indikators zurück, beim Misserfolg gibt [INVALID\\_HANDLE](#) zurück. Um Computerspeicher aus nicht genutzten Indikatoren zu befreien, wird die Funktion [IndicatorRelease\(\)](#) verwendet, zu deren Indikatorhandle passiert wird.

### Beispiel:

```
//+-----+
//|                                     Demo_iAMA.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
```

```

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Der Indikator zeigt, wie Daten erhalten werden sollen"
#property description "Indikator-Puffer für den technischen Indikator iAMA."
#property description "Symbol und Zeitrahmen, in denen der Indikator berechnet wird,"
#property description "sind durch die Parameter symbol und period angegeben."
#property description "Erstellungsweise des Handles ist durch den 'type' Parameter (T)"
#property description "Alle andere Parameter sind wie im normalen Indikator iAMA."

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots 1
//--- plot iAMA
#property indicator_label1 "iAMA"
#property indicator_type1  DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1 1
//+-----+
//| Aufzählung der Erstellungsweisen des Handles |
//+-----+
enum Creation
{
    Call_iAMA,          // iAMA verwenden
    Call_IndicatorCreate // IndicatorCreate verwenden
};
//--- Eingabeparameter
input Creation      type=Call_iAMA;          // Funktionstyp
input string        symbol=" ";              // Symbol
input ENUM_TIMEFRAMES period=PERIOD_CURRENT; // Zeitrahmen
input int           ama_period=15;           // Berechnungszeitraum
input int           fast_ma_period=2;        // Schnelle MA Periode
input int           slow_ma_period=30;       // Langsame MA Periode
input int           ama_shift=0;             // horizontale Verschiebung des I
input ENUM_APPLIED_PRICE applied_price;      // Preistyp
//--- Indikator-Puffer
double           iAMABuffer[];
//--- Eine Variable um Handle des Indikators iAMA zu speichern
int             handle;
//--- Variable für Speicherung
string name=symbol;
//--- Name des Indikators auf dem Chart
string short_name;
//--- die Anzahl der Werte im Indikator Adaptive Moving Average gespeichert wird
int             bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+

```

```

int OnInit()
{
    //--- Bindung von Array zum Indikator-Puffer
    SetIndexBuffer(0,iAMABuffer,INDICATOR_DATA);
    //--- Verschiebung definieren
    PlotIndexSetInteger(0,PLOT_SHIFT,ama_shift);
    //--- das Symbol, auf das der Indikator gebaut wird, definieren
    name=symbol;
    //--- Leerzeichen aus dem linken und rechten löschen
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- Wenn nach diesem die Länge des String name ist Null
    if(StringLen(name)==0)
    {
        //--- das Symbol aus dem Chart, auf dem der Indikator läuft, nehmen
        name=_Symbol;
    }
    //--- Erstellen ein Handel des Indikators
    if(type==Call_iAMA)
        handle=iAMA(name,period,ama_period,fast_ma_period,slow_ma_period,ama_shift,applied_price);
    else
    {
        //--- die Struktur mit Werten von Indikatorparametern ausfüllen
        MqlParam pars[5];
        pars[0].type=TYPE_INT;
        pars[0].integer_value=ama_period;
        pars[1].type=TYPE_INT;
        pars[1].integer_value=fast_ma_period;
        pars[2].type=TYPE_INT;
        pars[2].integer_value=slow_ma_period;
        pars[3].type=TYPE_INT;
        pars[3].integer_value=ama_shift;
        //--- Preistyp
        pars[4].type=TYPE_INT;
        pars[4].integer_value=applied_price;
        handle=IndicatorCreate(name,period,IND_AMA,5,pars);
    }
    //--- Wenn Handle konnte nicht erstellt werden
    if(handle==INVALID_HANDLE)
    {
        //--- darüber schreiben und Nummer des Fehlers anzeigen
        PrintFormat("Handle des Indikators iAMA für das Paar %s/%s konnte nicht erstellt werden",
                    name,
                    EnumToString(period),
                    GetLastError());
        //--- Arbeit des Indikators ist früher geendet
        return(INIT_FAILED);
    }
    //--- Das Paar, Symbol/Zeitraumen, auf deren Adaptive Moving Average berechnet war, zeichnen

```

```

    short_name=StringFormat("iAMA(%s/%s,%d,%d,%d,d)",name,EnumToString(period),ama_peri
    IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- normale Initialisierung des Indikators
    return(INIT_SUCCEEDED);
};
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{
//--- Anzahl der Werte des Indikators iAlligator zu kopieren
    int values_to_copy;
//--- Anzahl der berechneten Werte im Indikator finden
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() hat %d zurückgegeben, Fehlercode ist %d",calculated,GetLastError());
        return(0);
    };
//--- wenn dies der erste Start der Berechnung des Indikators ist oder die Anzahl der
//--- oder wenn Sie brauchen den Indikator für zwei oder mehr Balken (was bedeutet, da
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
    {
        //--- wenn Array iADBuffer größer als die Anzahl der Werte in iAD auf symbol/peri
        //--- sonst kopieren wir weniger als Größe der Indikator-Puffer
        if(calculated>rates_total) values_to_copy=rates_total;
        else values_to_copy=calculated;
    };
    else
    {
        //--- dann ist unser Indikator nicht das erste Mal berechnet, und seit dem letzten
        //--- nicht mehr als ein Balken für die Berechnung hinzugefügt war
        values_to_copy=(rates_total-prev_calculated)+1;
    };
//--- Die Arrays mit den Werten aus dem Indikator Alligator ausfüllen
//--- Wenn FillArraysFromBuffer false zurückgegeben hat, dann die Daten nicht bereit s
    if(!FillArrayFromBuffer(iAMABuffer,ama_shift,handle,values_to_copy)) return(0);
//--- Nachricht bilden
    string comm=StringFormat("%s ==> Aktualisierte Werte im Indikator %s: %d",
                             TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),

```

```

        short_name,
        values_to_copy);
//--- Hilfsnachrichtung auf dem Chart anzeigen
    Comment(comm);
//--- die Anzahl der Werte im Indikator Accelerator Oscillator speichern
    bars_calculated=calculated;
//--- den Wert prev_calculated für den nächsten Anruf zurückgeben
    return(rates_total);
};
//+-----+
//| Indikator-Puffer aus dem Indikator iAMA ausfüllen |
//+-----+
bool FillArrayFromBuffer(double &ama_buffer[], // Indikator-Puffer der Linie von AMA
                        int a_shift,           // Verschiebung der Linie AMA
                        int ind_handle,        // Handle von iAMA
                        int amount            // Anzahl der Werte, die kopiert werden
                        )
{
//--- Fehlercode rücksetzen
    ResetLastError();
//--- Teil des Arrays iAMABuffer mit Werten auf Indikator-Puffer mit Index 0 ausfüllen
    if(CopyBuffer(ind_handle,0,-a_shift,amount,ama_buffer)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iAlligator kopiert werden, Fehlercode: %d", GetLastError());
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden konnte
        return(false);
    }
//--- Alles gelang
    return(true);
};
//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
//--- Das Chart nach der Löschung des Indikators leeren
    Comment("");
};

```



## iAO

Gibt das Handle des Indikators Awesome Oscillator zurück. Hat nur einen Puffer.

```
int iAO(
    string      symbol,      // Symbolname
    ENUM_TIMEFRAMES period   // Periode
);
```

### Parameter

*symbol*

[in] Symbolname des Instrumentes, dessen Daten für Berechnung des Indikators verwendet werden. NULL bedeutet das laufende Symbol.

*period*

[in] Wert der Periode kann einer der Aufzählungswerte ENUM\_TIMEFRAMES sein, 0 bedeutet das laufende Timeframe.

### Rückgabewert

Gibt das Handle des angegebenen technischen Indikators zurück, beim Misserfolg gibt INVALID\_HANDLE zurück. Um Computerspeicher aus nicht genutzten Indikatoren zu befreien, wird die Funktion IndicatorRelease() verwendet, zu deren Indikatorhandle passiert wird.

### Beispiel:

```
//+-----+
//|                                     Demo_iAO.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Der Indikator zeigt, wie Daten erhalten werden sollen"
#property description "Indikator-Puffer für den technischen Indikator iAO."
#property description "Symbol und Zeitrahmen, in denen der Indikator berechnet wird,"
#property description "sind durch die Parameter symbol und period angegeben."
#property description "Erstellungsweise des Handles ist durch den 'type' Parameter (T)"

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots 1
//--- iAO bauen
#property indicator_label1 "iAO"
#property indicator_type1  DRAW_COLOR_HISTOGRAM
#property indicator_color1 clrGreen,clrRed
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//+-----+
```

```

//| Aufzählung der Erstellungsweisen des Handles |
//+-----+
enum Creation
{
    Call_iAO,           // iAO verwenden
    Call_IndicatorCreate // IndicatorCreate verwenden
};

//--- Eingabeparameter
input Creation      type=Call_iAO;           // Funktionstyp
input string        symbol=" ";              // Symbol
input ENUM_TIMEFRAMES period=PERIOD_CURRENT; // Zeitrahmen

//--- Indicator-Puffer
double      iAOBuffer[];
double      iAOCOLORS[];

//--- Eine Variable um Handle des Indikators iAO zu speichern
int  handle;

//--- Variable für Speicherung
string name=symbol;

//--- Name des Indikators auf dem Chart
string short_name;

//--- die Anzahl der Werte im Indikator Awesome Oscillator gespeichert wird
int  bars_calculated=0;

//+-----+
//| Custom indicator initialization function |
//+-----+

int OnInit()
{
    //--- Bindung von Arrays zu den Indikator-Puffern
    SetIndexBuffer(0,iAOBuffer,INDICATOR_DATA);
    SetIndexBuffer(1,iAOCOLORS,INDICATOR_COLOR_INDEX);

    //--- das Symbol, auf das der Indikator gebaut wird, definieren
    name=symbol;

    //--- Leerzeichen aus dem linken und rechten löschen
    StringTrimRight(name);
    StringTrimLeft(name);

    //--- Wenn nach diesem die Länge des String name ist Null
    if(StringLen(name)==0)
    {
        //--- das Symbol aus dem Chart, auf dem der Indikator läuft, nehmen
        name=_Symbol;
    }

    //--- Erstellen ein Handel des Indikators
    if(type==Call_iAO)
        handle=iAO(name,period);
    else
        handle=IndicatorCreate(name,period,IND_AO);

    //--- Wenn Handle konnte nicht erstellt werden
    if(handle==INVALID_HANDLE)
    {

```

```

    //--- darüber schreiben und Nummer des Fehlers anzeigen
    PrintFormat("Handle des Indikators iAO für das Paar %s/%s konnte nicht erstellt
        name,
        EnumToString(period),
        GetLastError());

    //--- Arbeit des Indikators ist früher geendet
    return(INIT_FAILED);
; }

//--- Das Paar, Symbol/Zeitraumen, auf deren Awesome Oscillator berechnet war, zeigen
short_name=StringFormat("iAO(%s/%s)",name,EnumToString(period));
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- normale Initialisierung des Indikators
return(INIT_SUCCEEDED);
; }

//+-----+
//| Custom indicator iteration function |
//+-----+

int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{
    //--- Anzahl der Werte des Indikators iAO zu kopieren
    int values_to_copy;
    //--- Anzahl der berechneten Werte im Indikator finden
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() hat %d zurückgegeben, Fehlercode ist %d",calculated,GetLastError());
        return(0);
    }
; }

//--- wenn dies der erste Start der Berechnung des Indikators ist oder die Anzahl der
//--- oder wenn Sie brauchen den Indikator für zwei oder mehr Balken (was bedeutet, dass
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
    {
        //--- wenn Array iAOBuffer größer als die Anzahl der Werte in iAO auf symbol/period
        //--- sonst kopieren wir weniger als Größe der Indikator-Puffer
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    }
; }
else
{
    //--- dann ist unser Indikator nicht das erste Mal berechnet, und seit dem letzten

```

```

        //--- nicht mehr als ein Balken für die Berechnung hinzugefügt war
        values_to_copy=(rates_total-prev_calculated)+1;
    };
//--- Arrays iAObuffer und iAOCcolors mit den Werten aus dem Indikator Awesome Oscillator
//--- Wenn FillArraysFromBuffer false zurückgegeben hat, dann die Daten nicht berechnen
    if(!FillArraysFromBuffer(iAObuffer,iAOCcolors,handle,values_to_copy)) return(0);
//--- Nachricht bilden
    string comm=StringFormat("%s ==> Aktualisierte Werte im Indikator %s: %d",
                              TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                              short_name,
                              values_to_copy);
//--- Hilfsnachrichtung auf dem Chart anzeigen
    Comment(comm);
//--- die Anzahl der Werte im Indikator Accelerator Oscillator speichern
    bars_calculated=calculated;
//--- den Wert prev_calculated für den nächsten Anruf zurückgeben
    return(rates_total);
};
//+-----+
//| Indikator-Puffer aus dem Indikator iAO ausfüllen |
//+-----+
bool FillArraysFromBuffer(double &values[], // Indikator-Puffer mit Werten von
                          double &color_indexes[], // Farbpuffer (um Farbindex zu speichern)
                          int ind_handle, // Handle von iAO
                          int amount // Anzahl der Werte, die kopiert werden
                          )
{
    //--- Fehlercode rücksetzen
    ResetLastError();
//--- Teil des Arrays iAObuffer mit Werten auf Indikator-Puffer mit Index 0 ausfüllen
    if(CopyBuffer(ind_handle,0,0,amount,values)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iAO kopiert werden, Fehlercode: %d",GetLastError());
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden konnte
        return(false);
    };
//--- Nun die Farbeindizes kopieren
    if(CopyBuffer(ind_handle,1,0,amount,color_indexes)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Die Farbwerte konnte nicht aus dem Indikator iAO kopiert werden, Fehlercode: %d",GetLastError());
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden konnte
        return(false);
    };
//--- Alles gelang
    return(true);
};
//+-----+

```

```
///| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
//--- Das Chart nach der Löschung des Indikators leeren
    Comment("");
};
```

## iATR

Gibt das Handle des Indikators Average True Range zurück. Hat nur einen Puffer.

```
int iATR(
    string      symbol,      // Symbolname
    ENUM_TIMEFRAMES period,  // Periode
    int         ma_period    // Glättungsperiode
);
```

### Parameter

*symbol*

[in] Symbolname des Instruments, dessen Daten für Berechnung des Indikators verwendet werden. NULL bedeutet das laufende Symbol.

*period*

[in] Wert der Periode kann einer der Aufzählungswerte ENUM\_TIMEFRAMES sein, 0 bedeutet das laufende Timeframe.

*ma\_period*

[in] Glättungsperiode für Berechnung des Indikators.

### Rueckgabewert

Gibt das Handle des angegebenen technischen Indikators zurück, beim Misserfolg gibt INVALID\_HANDLE zurück. Um Computerspeicher aus nicht genutzten Indikatoren zu befreien, wird die Funktion IndicatorRelease() verwendet, zu deren Indikatorhandle passiert wird.

### Beispiel:

```
//+-----+
//|                                     Demo_iATR.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Der Indikator zeigt, wie Daten erhalten werden sollen"
#property description "Indikator-Puffer für den technischen Indikator iATR."
#property description "Symbol und Zeitrahmen, in denen der Indikator berechnet wird,"
#property description "sind durch die Parameter symbol und period angegeben."
#property description "Erstellungsweise des Handles ist durch den 'type' Parameter (T)"

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- plot iATR
#property indicator_label1  "iATR"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
```

```

#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+-----+
//| Aufzählung der Erstellungsweisen des Handles |
//+-----+
enum Creation
{
    Call_iATR, // iATR verwenden
    Call_IndicatorCreate // IndicatorCreate verwenden
};
//--- Eingabeparameter
input int          atr_period=14;           // Berechnungszeitraum
input Creation     type=Call_iATR;          // Funktionstyp
input string       symbol=" ";              // Symbol
input ENUM_TIMEFRAMES period=PERIOD_CURRENT; // Zeitrahmen
//--- Indicator-Puffer
double            iATRBuffer[];
//--- Eine Variable um Handle des Indikators iAC zu speichern
int              handle;
//--- Variable für Speicherung
string name=symbol;
//--- Name des Indikators auf dem Chart
string short_name;
//--- die Anzahl der Werte im Indikator Average True Range gespeichert wird
int              bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- Bindung von Array zum Indikator-Puffer
    SetIndexBuffer(0,iATRBuffer,INDICATOR_DATA);
    //--- das Symbol, auf das der Indikator gebaut wird, definieren
    name=symbol;
    //--- Leerzeichen aus dem linken und rechten löschen
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- Wenn nach diesem die Länge des String name ist Null
    if(StringLen(name)==0)
    {
        //--- das Symbol aus dem Chart, auf dem der Indikator läuft, nehmen
        name=_Symbol;
    };
    //--- Erstellen ein Handel des Indikators
    if(type==Call_iATR)
        handle=iATR(name,period,atr_period);
    else
    {
        //--- die Struktur mit Werten von Indikatorparametern ausfüllen

```

```

    MqlParam pars[1];
    pars[0].type=TYPE_INT;
    pars[0].integer_value=atr_period;
    handle=IndicatorCreate(name,period,IND_ATR,1,pars);
; }
//--- Wenn Handle konnte nicht erstellt werden
if(handle==INVALID_HANDLE)
{
    //--- darüber schreiben und Nummer des Fehlers anzeigen
    PrintFormat("Handle des Indikators iATR für das Paar %s/%s konnte nicht erstellt
                name,
                EnumToString(period),
                GetLastError());
    //--- Arbeit des Indikators ist früher geendet
    return(INIT_FAILED);
; }
//--- Das Paar, Symbol/Zeitraumen, auf deren Average True Range berechnet war, zeigen
short_name=StringFormat("iATR(%s/%s, period=%d)",name,EnumToString(period),atr_peri
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- normale Initialisierung des Indikators
return(INIT_SUCCEEDED);
; }
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
    //--- Anzahl der Werte des Indikators iATR zu kopieren
    int values_to_copy;
    //--- Anzahl der berechneten Werte im Indikator finden
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() hat %d zurückgegeben, Fehlercode ist %d",calculate
        return(0);
    ; }
    //--- wenn dies der erste Start der Berechnung des Indikators ist oder die Anzahl der
    //--- oder wenn Sie brauchen den Indikator für zwei oder mehr Balken (was bedeutet, da
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
    {

```



```

    //--- wenn Array iATRBuffer größer als die Anzahl der Werte in iATR auf symbol/
    //--- sonst kopieren wir weniger als Größe der Indikator-Puffer
    if(calculated>rates_total) values_to_copy=rates_total;
    else
        values_to_copy=calculated;
    ;}
else
{
    //--- dann ist unser Indikator nicht das erste Mal berechnet, und seit dem letzten
    //--- nicht mehr als ein Balken für die Berechnung hinzugefügt war
    values_to_copy=(rates_total-prev_calculated)+1;
    ;}
//--- Das Array iATRBuffer mit den Werten aus dem Indikator Average True Range ausfüllen
//--- Wenn FillArrayFromBuffer false zurückgegeben hat, dann die Daten nicht bereit sein
    if(!FillArrayFromBuffer(iATRBuffer,handle,values_to_copy)) return(0);
//--- Nachricht bilden
    string comm=StringFormat("%s ==> Aktualisierte Werte im Indikator %s: %d",
                                TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                                short_name,
                                values_to_copy);
//--- Hilfsnachrichtung auf dem Chart anzeigen
    Comment(comm);
//--- die Anzahl der Werte im Indikator Accelerator Oscillator speichern
    bars_calculated=calculated;
//--- den Wert prev_calculated für den nächsten Anruf zurückgeben
    return(rates_total);
    ;}
//+-----+
//| Indikator-Puffer aus dem Indikator iATR ausfüllen |
//+-----+
bool FillArrayFromBuffer(double &values[], // Indikator-Puffer der Werte von ATR
                        int ind_handle,    // Handle von iATR
                        int amount        // Anzahl der Werte, die kopiert werden
                        )
{
    //--- Fehlercode rücksetzen
    ResetLastError();
//--- Teil des Arrays iATRBuffer mit Werten auf Indikator-Puffer mit Index 0 ausfüllen
    if(CopyBuffer(ind_handle,0,0,amount,values)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iATR kopiert werden, Fehlercode: %d",GetLastError());
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden konnte
        return(false);
    }
    ;}
//--- Alles gelang
    return(true);
    ;}
//+-----+
//| Indicator deinitialization function |

```

```
//+-----+  
void OnDeinit(const int reason)  
{  
    //--- Das Chart nach der Löschung des Indikators leeren  
    Comment("");  
};
```

## iBearsPower

Gibt das Handle des Indikators Bears Power zurück. Hat nur einen Puffer.

```
int iBearsPower(
    string      symbol,           // Symbolname
    ENUM_TIMEFRAMES period,      // Periode
    int         ma_period,       // Mittelungsperiode
);
```

### Parameter

*symbol*

[in] Symbolname des Instrumentes, dessen Daten für Berechnung des Indikators verwendet werden. NULL bedeutet das laufende Symbol.

*period*

[in] Wert der Periode kann einer der Aufzählungswerte ENUM\_TIMEFRAMES sein, 0 bedeutet das laufende Timeframe.

*ma\_period*

[in] Glättungsperiode für Berechnung des Indikators.

### Rückgabewert

Gibt das Handle des angegebenen technischen Indikators zurück, beim Misserfolg gibt INVALID\_HANDLE zurück. Um Computerspeicher aus nicht genutzten Indikatoren zu befreien, wird die Funktion IndicatorRelease() verwendet, zu deren Indikatorhandle passiert wird.

### Beispiel:

```
//+-----+
//|                                     Demo_iBearsPower.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Der Indikator zeigt, wie Daten erhalten werden sollen"
#property description "Indikator-Puffer für den technischen Indikator iBearsPower."
#property description "Symbol und Zeitrahmen, in denen der Indikator berechnet wird,"
#property description "sind durch die Parameter symbol und period angegeben."
#property description "Erstellungsweise des Handles ist durch den 'type' Parameter (T)"

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- iBearsPower bauen
#property indicator_label1  "iBearsPower"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrSilver
```

```

#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+-----+
//| Aufzählung der Erstellungsweisen des Handles |
//+-----+
enum Creation
{
    Call_iBearsPower,      // iBearsPower verwenden
    Call_IndicatorCreate   // IndicatorCreate verwenden
};
//--- Eingabeparameter
input Creation          type=Call_iBearsPower;  // Funktionstyp
input int               ma_period=13;           // MA Periode
input string            symbol=" ";             // Symbol
input ENUM_TIMEFRAMES   period=PERIOD_CURRENT; // Zeitrahmen
//--- Indicator-Puffer
double                iBearsPowerBuffer[];
//--- Eine Variable um Handle des Indikators iBearsPower zu speichern
int    handle;
//--- Variable für Speicherung
string name=symbol;
//--- Name des Indikators auf dem Chart
string short_name;
//--- die Anzahl der Werte im Indikator Bears Power gespeichert wird
int    bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- Bindung von Array zum Indikator-Puffer
    SetIndexBuffer(0,iBearsPowerBuffer,INDICATOR_DATA);
    //--- das Symbol, auf das der Indikator gebaut wird, definieren
    name=symbol;
    //--- Leerzeichen aus dem linken und rechten löschen
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- Wenn nach diesem die Länge des String name ist Null
    if(StringLen(name)==0)
    {
        //--- das Symbol aus dem Chart, auf dem der Indikator läuft, nehmen
        name=_Symbol;
    }
    //--- Erstellen ein Handel des Indikators
    if(type==Call_iBearsPower)
        handle=iBearsPower(name,period,ma_period);
    else
    {
        //--- die Struktur mit Werten von Indikatorparametern ausfüllen

```

```

MqlParam pars[1];
//--- MA Periode
pars[0].type=TYPE_INT;
pars[0].integer_value=ma_period;
handle=IndicatorCreate(name,period,IND_BEARS,1,pars);
; }
//--- Wenn Handle konnte nicht erstellt werden
if(handle==INVALID_HANDLE)
{
    //--- darüber schreiben und Nummer des Fehlers anzeigen
    PrintFormat("Handle des Indikators iBearsPower für das Paar %s/%s konnte nicht e
        name,
        EnumToString(period),
        GetLastError());
    //--- Arbeit des Indikators ist früher geendet
    return(INIT_FAILED);
; }
//--- Das Paar, Symbol/Zeitraumen, auf deren Bears Power berechnet war, zeigen
short_name=StringFormat("iBearsPower(%s/%s, period=%d)",name,EnumToString(period),p
    IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- normale Initialisierung des Indikators
return(INIT_SUCCEEDED);
; }
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{
    //--- Anzahl der Werte des Indikators iBearsPower zu kopieren
    int values_to_copy;
    //--- Anzahl der berechneten Werte im Indikator finden
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() hat %d zurückgegeben, Fehlercode ist %d",calculated,GetLastError());
        return(0);
    ; }
    //--- wenn dies der erste Start der Berechnung des Indikators ist oder die Anzahl der
    //--- oder wenn Sie brauchen den Indikator für zwei oder mehr Balken (was bedeutet, da
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated

```

```

    {
        //--- wenn Array iATRBuffer größer als die Anzahl der Werte in iBearsPower auf s
        //--- sonst kopieren wir weniger als Größe der Indikator-Puffer
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    };
else
    {
        //--- dann ist unser Indikator nicht das erste Mal berechnet, und seit dem letzte
        //--- nicht mehr als ein Balken für die Berechnung hinzugefügt war
        values_to_copy=(rates_total-prev_calculated)+1;
    };

//--- Das Array mit den Werten aus dem Indikator Bears Power ausfüllen
//--- Wenn FillArrayFromBuffer false zurückgegeben hat, dann die Daten nicht bereit s
    if(!FillArrayFromBuffer(iBearsPowerBuffer,handle,values_to_copy)) return(0);
//--- Nachricht bilden
    string comm=StringFormat("%s ==> Aktualisierte Werte im Indikator %s: %d",
                                TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                                short_name,
                                values_to_copy);

//--- Hilfsnachrichtung auf dem Chart anzeigen
    Comment(comm);

//--- die Anzahl der Werte im Indikator Bears Power speichern
    bars_calculated=calculated;
//--- den Wert prev_calculated für den nächsten Anruf zurückgeben
    return(rates_total);
};

//+-----+
//| Indikator-Puffer aus dem Indikator iBearsPower ausfüllen |
//+-----+

bool FillArrayFromBuffer(double &values[], // Indikator-Puffer der Werte von Bears Po
                        int ind_handle,    // Handle von iBearsPower
                        int amount         // Anzahl der Werte, die kopiert werden
                        )
{
    //--- Fehlercode rücksetzen
    ResetLastError();
    //--- Teil des Arrays iBearsPowerBuffer mit Werten auf Indikator-Puffer mit Index 0 au
    if(CopyBuffer(ind_handle,0,0,amount,values)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iBearsPower kopiert werden, Fe
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berech
        return(false);
    };
//--- Alles gelang
    return(true);
};

//+-----+

```

```
///| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
//--- Das Chart nach der Löschung des Indikators leeren
    Comment("");
};
```

## iBands

Gibt das Handle des Indikators Bollinger Bands® zurück.

```
int iBands(
    string          symbol,          // Symbolname
    ENUM_TIMEFRAMES period,          // Periode
    int             bands_period,     // Periode für Berechnung der Mittellinie
    int             bands_shift,      // horizontale Verschiebung des Indikators
    double          deviation,        //Anzahl der Standardabweichungen
    ENUM_APPLIED_PRICE applied_price  // Preistyp oder Handle
);
```

### Parameter

*symbol*

[in] Symbolname des Instrumentes, dessen Daten für Berechnung des Indikators verwendet werden. [NULL](#) bedeutet das laufende Symbol.

*period*

[in] Wert der Periode kann einer der Aufzählungswerte [ENUM\\_TIMEFRAMES](#) sein, 0 bedeutet das laufende Timeframe.

*bands\_period*

[in] Mittelungsperiode der Grundlinie des Indikators.

*bands\_shift*

[in] Verschiebung des Indikators in Bezug auf das Preischart.

*deviation*

[in] Abweichung von der Grundlinie.

*applied\_price*

[in] Der verwendete Preis. Kann eine der Preiskonstanten [ENUM\\_APPLIED\\_PRICE](#) oder Handle eines anderen Indikators sein.

### Rückgabewert

Gibt das Handle des angegebenen technischen Indikators zurück, beim Misserfolg gibt [INVALID\\_HANDLE](#) zurück. Um Computerspeicher aus nicht genutzten Indikatoren zu befreien, wird die Funktion [IndicatorRelease\(\)](#) verwendet, zu deren Indikatorhandle passiert wird.

### Bemerkung

Puffernummern: 0 - BASE\_LINE, 1 - UPPER\_BAND, 2 - LOWER\_BAND

### Beispiel:

```
//+-----+
//|                                     Demo_iBands.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
```



```

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Der Indikator zeigt, wie Daten erhalten werden sollen"
#property description "Indikator-Puffer für den technischen Indikator iBands."
#property description "Symbol und Zeitrahmen, in denen der Indikator berechnet wird,"
#property description "sind durch die Parameter symbol und period angegeben."
#property description "Erstellungsweise des Handles ist durch den 'type' Parameter (T)

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   3
//--- Upper bauen
#property indicator_label1  "Upper"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrMediumSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- Lower bauen
#property indicator_label2  "Lower"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrMediumSeaGreen
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- Middle bauen
#property indicator_label3  "Middle"
#property indicator_type3   DRAW_LINE
#property indicator_color3  clrMediumSeaGreen
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//+-----+
//| Aufzählung der Erstellungsweisen des Handles |
//+-----+
enum Creation
{
    Call_iBands,          // iBands verwenden
    Call_IndicatorCreate  // IndicatorCreate verwenden
};
//--- Eingabeparameter
input Creation          type=Call_iBands;          // Funktionstyp
input int               bands_period=20;           // MA Periode
input int               bands_shift=0;             // Verschiebung
input double            deviation=2.0;             // Anzahl der Standardabweichung
input ENUM_APPLIED_PRICE applied_price=PRICE_CLOSE; // Preistyp
input string            symbol=" ";               // Symbol
input ENUM_TIMEFRAMES   period=PERIOD_CURRENT;    // Zeitrahmen
//--- Indikator-Puffer
double      UpperBuffer[];
double      LowerBuffer[];

```

```

double      MiddleBuffer[];
//--- Eine Variable um Handle des Indikators iBands zu speichern
int      handle;
//--- Variable für Speicherung
string name=symbol;
//--- Name des Indikators auf dem Chart
string short_name;
//--- die Anzahl der Werte im Indikator Bollinger Bands gespeichert wird
int      bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- Bindung von Arrays zu den Indikator-Puffern
SetIndexBuffer(0,UpperBuffer,INDICATOR_DATA);
SetIndexBuffer(1,LowerBuffer,INDICATOR_DATA);
SetIndexBuffer(2,MiddleBuffer,INDICATOR_DATA);
//--- Verschiebung jede Linie definieren
PlotIndexSetInteger(0,PLOT_SHIFT,bands_shift);
PlotIndexSetInteger(1,PLOT_SHIFT,bands_shift);
PlotIndexSetInteger(2,PLOT_SHIFT,bands_shift);
//--- das Symbol, auf das der Indikator gebaut wird, definieren
name=symbol;
//--- Leerzeichen aus dem linken und rechten löschen
StringTrimRight(name);
StringTrimLeft(name);
//--- Wenn nach diesem die Länge des String name ist Null
if(StringLen(name)==0)
{
//--- das Symbol aus dem Chart, auf dem der Indikator läuft, nehmen
name=_Symbol;
};
//--- Erstellen ein Handel des Indikators
if(type==Call_iBands)
handle=iBands(name,period,bands_period,bands_shift,deviation,applied_price);
else
{
//--- die Struktur mit Werten von Indikatorparametern ausfüllen
MqlParam pars[4];
//--- MA Periode
pars[0].type=TYPE_INT;
pars[0].integer_value=bands_period;
//--- Verschiebung
pars[1].type=TYPE_INT;
pars[1].integer_value=bands_shift;
//--- Anzahl der Standardabweichungen
pars[2].type=TYPE_DOUBLE;
pars[2].double_value=deviation;

```

```

    //--- Preistyp
    pars[3].type=TYPE_INT;
    pars[3].integer_value=applied_price;
    handle=IndicatorCreate(name,period,IND_BANDS,4,pars);
    ;}
//--- Wenn Handle konnte nicht erstellt werden
if(handle==INVALID_HANDLE)
{
    //--- darüber schreiben und Nummer des Fehlers anzeigen
    PrintFormat("Handle des Indikators iBands für das Paar %s/%s konnte nicht erstellt werden",
        name,
        EnumToString(period),
        GetLastError());
    //--- Arbeit des Indikators ist früher geendet
    return(INIT_FAILED);
    ;}
//--- Das Paar, Symbol/Zeitraumen, auf deren Bollinger Bands berechnet war, zeigen
short_name=StringFormat("iBands(%s/%s, %d,%d,%G,%s)",name,EnumToString(period),
    bands_period,bands_shift,deviation,EnumToString(applied_price));
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- normale Initialisierung des Indikators
return(INIT_SUCCEEDED);
; }
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
    const int prev_calculated,
    const datetime &time[],
    const double &open[],
    const double &high[],
    const double &low[],
    const double &close[],
    const long &tick_volume[],
    const long &volume[],
    const int &spread[])
{
    //--- Anzahl der Werte des Indikators iBands zu kopieren
    int values_to_copy;
    //--- Anzahl der berechneten Werte im Indikator finden
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() hat %d zurückgegeben, Fehlercode ist %d",calculated,GetLastError());
        return(0);
    }
    ;}
//--- wenn dies der erste Start der Berechnung des Indikators ist oder die Anzahl der
//--- oder wenn Sie brauchen den Indikator für zwei oder mehr Balken (was bedeutet, dass
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)

```

```

    {
        //--- wenn die Größe der Indikatorsarrays ist größer als die Anzahl der Werte in
        //--- sonst kopieren wir weniger als Größe der Indikator-Puffer
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    };
else
    {
        //--- dann ist unser Indikator nicht das erste Mal berechnet, und seit dem letzten
        //--- nicht mehr als ein Balken für die Berechnung hinzugefügt war
        values_to_copy=(rates_total-prev_calculated)+1;
    };

//--- Das Array mit den Werten aus dem Indikator Bollinger Bands ausfüllen
//--- Wenn FillArraysFromBuffer false zurückgegeben hat, dann die Daten nicht bereit
if(!FillArraysFromBuffers(MiddleBuffer,UpperBuffer,LowerBuffer,bands_shift,handle,values_to_copy))
//--- Nachricht bilden
    string comm=StringFormat("%s ==> Aktualisierte Werte im Indikator %s: %d",
                                TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                                short_name,
                                values_to_copy);

//--- Hilfsnachrichtung auf dem Chart anzeigen
    Comment(comm);

//--- die Anzahl der Werte im Indikator Bollinger Bands speichern
    bars_calculated=calculated;

//--- den Wert prev_calculated für den nächsten Anruf zurückgeben
    return(rates_total);
};

//+-----+
//| Indikator-Puffer aus dem Indikator iBands ausfüllen |
//+-----+

bool FillArraysFromBuffers(double &base_values[], // Indikator-Puffer der Mittellinie
                           double &upper_values[], // Indikator-Puffer der Obergrenze
                           double &lower_values[], // Indikator-Puffer der Untergrenze
                           int shift, // Verschiebung
                           int ind_handle, // Handle von iBands
                           int amount // Anzahl der Werte, die kopiert werden
                           )
{
    //--- Fehlercode rücksetzen
    ResetLastError();

    //--- Teil des Arrays MiddleBuffer mit Werten auf Indikator-Puffer mit Index 0 ausfüllen
    if(CopyBuffer(ind_handle,0,-shift,amount,base_values)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iBands kopiert werden, Fehlercode: %d",GetLastError());
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden konnte
        return(false);
    };
};

```

```

//--- Teil des Arrays UpperBuffer mit Werten auf Indikator-Puffer mit Index 1 ausfüllen
if(CopyBuffer(ind_handle,1,-shift,amount,upper_values)<0)
{
    //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
    PrintFormat("Daten konnte nicht aus dem Indikator iBands kopiert werden, Fehlercode: %d", GetLastError());
    //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden konnte
    return(false);
};

//--- Teil des Arrays LowerBuffer mit Werten auf Indikator-Puffer mit Index 2 ausfüllen
if(CopyBuffer(ind_handle,2,-shift,amount,lower_values)<0)
{
    //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
    PrintFormat("Daten konnte nicht aus dem Indikator iBands kopiert werden, Fehlercode: %d", GetLastError());
    //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden konnte
    return(false);
};

//--- Alles gelang
return(true);
};

//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
    //--- Das Chart nach der Löschung des Indikators leeren
    Comment("");
};

```

## iBullsPower

Gibt das Handle des Indikators Bulls Power zurück. Hat nur einen Puffer.

```
int iBullsPower(
    string          symbol,          // Symbolname
    ENUM_TIMEFRAMES period,          // Periode
    int             ma_period,        // Mittelungsperiode
);
```

### Parameter

*symbol*

[in] Symbolname des Instrumentes, dessen Daten für Berechnung des Indikators verwendet werden. NULL bedeutet das laufende Symbol.

*period*

[in] Wert der Periode kann einer der Aufzählungswerte ENUM\_TIMEFRAMES, 0 bedeutet das laufende Timeframe.

*ma\_period*

[in] Mittelungsperiode für Berechnung des Indikators.

### Rückgabewert

Gibt das Handle des angegebenen technischen Indikators zurück, beim Misserfolg gibt INVALID\_HANDLE zurück. Um Computerspeicher aus nicht genutzten Indikatoren zu befreien, wird die Funktion IndicatorRelease() verwendet, zu deren Indikatorhandle passiert wird.

### Beispiel:

```
//+-----+
//|                                     Demo_iBullsPower.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Der Indikator zeigt, wie Daten erhalten werden sollen"
#property description "Indikator-Puffer für den technischen Indikator iBullsPower."
#property description "Symbol und Zeitrahmen, in denen der Indikator berechnet wird,"
#property description "sind durch die Parameter symbol und period angegeben."
#property description "Erstellungsweise des Handles ist durch den 'type' Parameter (T)"

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- iBullsPower bauen
#property indicator_label1   "iBullsPower"
#property indicator_type1    DRAW_HISTOGRAM
#property indicator_color1   clrSilver
```

```

#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+-----+
//| Aufzählung der Erstellungsweisen des Handles |
//+-----+
enum Creation
{
    Call_iBullsPower,      // iBullsPower verwenden
    Call_IndicatorCreate    // IndicatorCreate verwenden
};
//--- Eingabeparameter
input Creation      type=Call_iBullsPower;  // Funktionstyp
input int           ma_period=13;           // MA Periode
input string        symbol=" ";             // Symbol
input ENUM_TIMEFRAMES period=PERIOD_CURRENT; // Zeitrahmen
//--- Indicator-Puffer
double             iBullsPowerBuffer[];
//--- Eine Variable um Handle des Indikators iBullsPower zu speichern
int    handle;
//--- Variable für Speicherung
string name=symbol;
//--- Name des Indikators auf dem Chart
string short_name;
//--- die Anzahl der Werte im Indikator Bulls Power gespeichert wird
int    bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- Bindung von Array zum Indikator-Puffer
    SetIndexBuffer(0,iBullsPowerBuffer,INDICATOR_DATA);
    //--- das Symbol, auf das der Indikator gebaut wird, definieren
    name=symbol;
    //--- Leerzeichen aus dem linken und rechten löschen
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- Wenn nach diesem die Länge des String name ist Null
    if(StringLen(name)==0)
    {
        //--- das Symbol aus dem Chart, auf dem der Indikator läuft, nehmen
        name=_Symbol;
    }
    //--- Erstellen ein Handel des Indikators
    if(type==Call_iBullsPower)
        handle=iBullsPower(name,period,ma_period);
    else
    {
        //--- die Struktur mit Werten von Indikatorparametern ausfüllen

```

```

    MqlParam pars[1];
    //--- MA Periode
    pars[0].type=TYPE_INT;
    pars[0].integer_value=ma_period;
    handle=IndicatorCreate(name,period,IND_BULLS,1,pars);
; }
//--- Wenn Handle konnte nicht erstellt werden
if(handle==INVALID_HANDLE)
{
    //--- darüber schreiben und Nummer des Fehlers anzeigen
    PrintFormat("Handle des Indikators iBullsPower für das Paar %s/%s konnte nicht e
        name,
        EnumToString(period),
        GetLastError());
    //--- Arbeit des Indikators ist früher geendet
    return(INIT_FAILED);
; }
//--- Das Paar, Symbol/Zeitraumen, auf deren Bulls Power berechnet war, zeigen
short_name=StringFormat("iBullsPower(%s/%s, period=%d)",name,EnumToString(period),p
    IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- normale Initialisierung des Indikators
return(INIT_SUCCEEDED);
; }
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{
    //--- Anzahl der Werte des Indikators iBullsPower zu kopieren
    int values_to_copy;
    //--- Anzahl der berechneten Werte im Indikator finden
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() hat %d zurückgegeben, Fehlercode ist %d",calculated,GetLastError());
        return(0);
    ; }
    //--- wenn dies der erste Start der Berechnung des Indikators ist oder die Anzahl der
    //--- oder wenn Sie brauchen den Indikator für zwei oder mehr Balken (was bedeutet, da
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated

```



```

    {
        //--- wenn Array iBullsPowerBuffer größer als die Anzahl der Werte in iBullsPower
        //--- sonst kopieren wir weniger als Größe der Indikator-Puffer
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    };
else
    {
        //--- dann ist unser Indikator nicht das erste Mal berechnet, und seit dem letzten Mal
        //--- nicht mehr als ein Balken für die Berechnung hinzugefügt war
        values_to_copy=(rates_total-prev_calculated)+1;
    };
//--- Das Array iBullsPowerBuffer mit den Werten aus dem Indikator Bulls Power ausfüllen
//--- Wenn FillArrayFromBuffer false zurückgegeben hat, dann die Daten nicht bereit sein
if(!FillArrayFromBuffer(iBullsPowerBuffer,handle,values_to_copy)) return(0);
//--- Nachricht bilden
string comm=StringFormat("%s ==> Aktualisierte Werte im Indikator %s: %d",
                           TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                           short_name,
                           values_to_copy);
//--- Hilfsnachrichtung auf dem Chart anzeigen
Comment(comm);
//--- die Anzahl der Werte im Indikator Bulls Power speichern
bars_calculated=calculated;
//--- den Wert prev_calculated für den nächsten Anruf zurückgeben
return(rates_total);
};
//+-----+
//| Indikator-Puffer aus dem Indikator iBullsPower ausfüllen |
//+-----+
bool FillArrayFromBuffer(double &values[], // Indikator-Puffer der Werte von Bulls Power
                        int ind_handle,    // Handle von iBullsPower
                        int amount        // Anzahl der Werte, die kopiert werden
                        )
{
    //--- Fehlercode rücksetzen
    ResetLastError();
    //--- Teil des Arrays iBullsPowerBuffer mit Werten auf Indikator-Puffer mit Index 0 ausfüllen
    if(CopyBuffer(ind_handle,0,0,amount,values)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iBullsPower kopiert werden, Fehlercode: %d",
                    GetLastError());
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet wurde
        return(false);
    };
//--- Alles gelang
return(true);
};
//+-----+

```

```
///| Indicator deinitialization function |  
//+-----+  
void OnDeinit(const int reason)  
{  
//--- Das Chart nach der Löschung des Indikators leeren  
    Comment("");  
};  
//+-----+
```

## iCCI

Gibt das Handle des Indikators Commodity Channel Index zurück. Hat nur einen Puffer.

```
int iCCI(
    string          symbol,          // Symbolname
    ENUM_TIMEFRAMES period,          // Periode
    int             ma_period,        // Mittelungsperiode
    ENUM_APPLIED_PRICE applied_price // Preistyp oder Handle
);
```

### Parameter

*symbol*

[in] Symbolname des Instrumentes, dessen Daten für Berechnung des Indikators verwendet werden. NULL bedeutet das laufende Symbol.

*period*

[in] Wert der Periode kann einer der Aufzählungswerte ENUM\_TIMEFRAMES sein, 0 bedeutet das laufende Timeframe.

*ma\_period*

[in] Mittelungsperiode für Berechnung des Indikators.

*applied\_price*

[in] Der verwendete Preis. Kann eine der Preiskonstanten ENUM\_APPLIED\_PRICE oder handle eines anderen Indikators sein.

### Rückgabewert

Gibt das Handle des angegebenen technischen Indikators zurück, beim Misserfolg gibt INVALID\_HANDLE zurück. Um Computerspeicher aus nicht genutzten Indikatoren zu befreien, wird die Funktion IndicatorRelease() verwendet, zu deren Indikatorhandle passiert wird.

### Beispiel:

```
//+-----+
//|                                     Demo_iCCI.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Der Indikator zeigt, wie Daten erhalten werden sollen"
#property description "Indikator-Puffer für den technischen Indikator iCCI."
#property description "Symbol und Zeitrahmen, in denen der Indikator berechnet wird,"
#property description "sind durch die Parameter symbol und period angegeben."
#property description "Erstellungsweise des Handles ist durch den 'type' Parameter (T)"

#property indicator_separate_window
#property indicator_buffers 1
```

```

#property indicator_plots 1
//--- iCCI bauen
#property indicator_label1 "iCCI"
#property indicator_type1 DRAW_LINE
#property indicator_color1 clrLightSeaGreen
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//--- Horizontale Ebenen im Indikatorsfenster
#property indicator_level1 -100.0
#property indicator_level2 100.0
//+-----+
//| Aufzählung der Erstellungsweisen des Handles |
//+-----+
enum Creation
{
    Call_iCCI,          // iCCI verwenden
    Call_IndicatorCreate // IndicatorCreate verwenden
};
//--- Eingabeparameter
input Creation      type=Call_iCCI;          // Funktionstyp
input int           ma_period=14;            // MA Periode
input ENUM_APPLIED_PRICE applied_price=PRICE_TYPICAL; // Preistyp
input string        symbol=" ";              // Symbol
input ENUM_TIMEFRAMES period=PERIOD_CURRENT; // Zeitrahmen
//--- Indicator-Puffer
double iCCIBuffer[];
//--- Eine Variable um Handle des Indikators iCCI zu speichern
int handle;
//--- Variable für Speicherung
string name=symbol;
//--- Name des Indikators auf dem Chart
string short_name;
//--- die Anzahl der Werte im Indikator Commodity Channel Index gespeichert wird
int bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- Bindung von Array zum Indikator-Puffer
    SetIndexBuffer(0,iCCIBuffer,INDICATOR_DATA);
    //--- das Symbol, auf das der Indikator gebaut wird, definieren
    name=symbol;
    //--- Leerzeichen aus dem linken und rechten löschen
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- Wenn nach diesem die Länge des String name ist Null
    if(StringLen(name)==0)
    {

```

```

    //--- das Symbol aus dem Chart, auf dem der Indikator läuft, nehmen
    name=_Symbol;
};

//--- Erstellen ein Handel des Indikators
if(type==Call_iCCI)
    handle=iCCI(name,period,ma_period,applied_price);
else
{
    //--- die Struktur mit Werten von Indikatorparametern ausfüllen
    MqlParam pars[2];
    //--- MA Periode
    pars[0].type=TYPE_INT;
    pars[0].integer_value=ma_period;
    //--- Preistyp
    pars[1].type=TYPE_INT;
    pars[1].integer_value=applied_price;
    handle=IndicatorCreate(name,period,IND_CCI,2,pars);
};

//--- Wenn Handle konnte nicht erstellt werden
if(handle==INVALID_HANDLE)
{
    //--- darüber schreiben und Nummer des Fehlers anzeigen
    PrintFormat("Handle des Indikators iCCI für das Paar %s/%s konnte nicht erstellt
                name,
                EnumToString(period),
                GetLastError());

    //--- Arbeit des Indikators ist früher geendet
    return(INIT_FAILED);
};

//--- Das Paar, Symbol/Zeitrahen, auf deren Bollinger Bands® berechnet war, zeigen
short_name=StringFormat("iCCI(%s/%s, %d, %s)",name,EnumToString(period),
                        ma_period,EnumToString(applied_price));

IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- normale Initialisierung des Indikators
return(INIT_SUCCEEDED);
};

//+-----+
//| Custom indicator iteration function |
//+-----+

int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])

```

```

{
//--- Anzahl der Werte des Indikators iCCI zu kopieren
    int values_to_copy;
//--- Anzahl der berechneten Werte im Indikator finden
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() hat %d zurückgegeben, Fehlercode ist %d",calculated,GetLastError());
        return(0);
    }
//--- wenn dies der erste Start der Berechnung des Indikators ist oder die Anzahl der
//--- oder wenn Sie brauchen den Indikator für zwei oder mehr Balken (was bedeutet, dass
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
    {
        //--- wenn Array iATRBuffer größer als die Anzahl der Werte in iCCI auf symbol/price
        //--- sonst kopieren wir weniger als Größe der Indikator-Puffer
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    }
else
    {
        //--- dann ist unser Indikator nicht das erste Mal berechnet, und seit dem letzten
        //--- nicht mehr als ein Balken für die Berechnung hinzugefügt war
        values_to_copy=(rates_total-prev_calculated)+1;
    }
//--- Das Array iCCIBuffer mit den Werten aus dem Indikator Commodity Channel Index ausfüllen
//--- Wenn FillArrayFromBuffer false zurückgegeben hat, dann die Daten nicht bereit sind
    if(!FillArrayFromBuffer(iCCIBuffer,handle,values_to_copy)) return(0);
//--- Nachricht bilden
    string comm=StringFormat("%s ==> Aktualisierte Werte im Indikator %s: %d",
                               TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                               short_name,
                               values_to_copy);
//--- Hilfsnachrichtung auf dem Chart anzeigen
    Comment(comm);
//--- die Anzahl der Werte im Indikator Commodity Channel Index speichern
    bars_calculated=calculated;
//--- den Wert prev_calculated für den nächsten Anruf zurückgeben
    return(rates_total);
};

//+-----+
//| Indikator-Puffer aus dem Indikator iCCI ausfüllen |
//+-----+
bool FillArrayFromBuffer(double &values[], // Indikator-Puffer der Werte von Commodity Channel Index
                        int ind_handle, // Handle von iCCI
                        int amount // Anzahl der Werte, die kopiert werden
                        )
{
//--- Fehlercode rücksetzen

```

```

    ResetLastError();
    ///--- Teil des Arrays iCCIBuffer mit Werten auf Indikator-Puffer mit Index 0 ausfüllen
    if(CopyBuffer(ind_handle,0,0,amount,values)<0)
    {
        ///--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iCCI kopiert werden, Fehlercode: %d", GetLastError());
        ///--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden konnte
        return(false);
    }
    ///--- Alles gelang
    return(true);
};

//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
    ///--- Das Chart nach der Löschung des Indikators leeren
    Comment("");
};

```

## iChaikin

Gibt das Handle des Indikators Chaikin Oscillator zurück. Hat nur einen Puffer.

```
int iChaikin(
    string          symbol,          // Symbolname
    ENUM_TIMEFRAMES period,          // Periode
    int             fast_ma_period,  // schnelle Periode
    int             slow_ma_period,  // langsame Periode
    ENUM_MA_METHOD  ma_method,       // Glättungstyp
    ENUM_APPLIED_VOLUME applied_volume // das verwendete Volumen
);
```

### Parameter

*symbol*

[in] Symbolname des Instrumentes, dessen Daten für Berechnung des Indikators verwendet werden. [NULL](#) bedeutet das laufende Symbol.

*period*

[in] Wert der Periode kann einer der Aufzählungswerte [ENUM\\_TIMEFRAMES](#) sein, 0 bedeutet das laufende Timeframe.

*fast\_ma\_period*

[in] Schnelle Mittelungsperiode für Berechnung des Indikators.

*slow\_ma\_period*

[in] langsame Mittelungsperiode für Berechnung des Indikators .

*ma\_method*

[in] Glättungstyp .Kann eine der Glättungskonstanten von [ENUM\\_MA\\_METHOD](#) sein.

*applied\_volume*

[in] Das verwendete Volumen. Kann eine der Aufzählungskonstanten [ENUM\\_APPLIED\\_VOLUME](#) sein.

### Rückgabewert

Gibt das Handle des angegebenen technischen Indikators zurück, beim Misserfolg gibt [INVALID\\_HANDLE](#) zurück. Um Computerspeicher aus nicht genutzten Indikatoren zu befreien, wird die Funktion [IndicatorRelease\(\)](#) verwendet, zu deren Indikatorhandle passiert wird.

### Beispiel:

```
//+-----+
//|                                     Demo_iChaikin.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Der Indikator zeigt, wie Daten erhalten werden sollen"
```



```

#property description "Indikator-Puffer für den technischen Indikator iChaikin."
#property description "Symbol und Zeitrahmen, in denen der Indikator berechnet wird,"
#property description "sind durch die Parameter symbol und period angegeben."
#property description "Erstellungsweise des Handles ist durch den 'type' Parameter (Ty

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots 1
//--- iChaikin bauen
#property indicator_label1 "iChaikin"
#property indicator_type1 DRAW_LINE
#property indicator_color1 clrLightSeaGreen
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//+-----+
//| Aufzählung der Erstellungsweisen des Handles |
//+-----+
enum Creation
{
    Call_iChaikin,          // iChaikin verwenden
    Call_IndicatorCreate    // IndicatorCreate verwenden
};
//--- Eingabeparameter
input Creation          type=Call_iChaikin;          // Funktionstyp
input int               fast_ma_period=3;            // Schnelle Zeitraum
input int               slow_ma_period=10;           // Langsame Zeitraum
input ENUM_MA_METHOD    ma_method=MODE_EMA;         // Art der Glättung
input ENUM_APPLIED_VOLUME applied_volume=VOLUME_TICK; // Volumentyp
input string            symbol=" ";                  // Symbol
input ENUM_TIMEFRAMES   period=PERIOD_CURRENT;      // Zeitrahmen
//--- Indikator-Puffer
double iChaikinBuffer[];
//--- Eine Variable um Handle des Indikators iChaikin zu speichern
int handle;
//--- Variable für Speicherung
string name=symbol;
//--- Name des Indikators auf dem Chart
string short_name;
//--- die Anzahl der Werte im Indikator Chaikin Oscillator gespeichert wird
int bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- Bindung von Array zum Indikator-Puffer
    SetIndexBuffer(0,iChaikinBuffer,INDICATOR_DATA);
    //--- das Symbol, auf das der Indikator gebaut wird, definieren
    name=symbol;

```

```

//--- Leerzeichen aus dem linken und rechten löschen
StringTrimRight(name);
StringTrimLeft(name);
//--- Wenn nach diesem die Länge des String name ist Null
if(StringLen(name)==0)
{
    //--- das Symbol aus dem Chart, auf dem der Indikator läuft, nehmen
    name=_Symbol;
};
//--- Erstellen ein Handel des Indikators
if(type==Call_iChaikin)
    handle=iChaikin(name,period,fast_ma_period,slow_ma_period,ma_method,applied_volume);
else
{
    //--- die Struktur mit Werten von Indikatorparametern ausfüllen
    MqlParam pars[4];
    //--- Schnelle Zeitraum
    pars[0].type=TYPE_INT;
    pars[0].integer_value=fast_ma_period;
    //--- Langsame Periode
    pars[1].type=TYPE_INT;
    pars[1].integer_value=slow_ma_period;
    //--- Art der Glättung
    pars[2].type=TYPE_INT;
    pars[2].integer_value=ma_method;
    //--- Volumentyp
    pars[3].type=TYPE_INT;
    pars[3].integer_value=applied_volume;
    handle=IndicatorCreate(name,period,IND_CHAIKIN,4,pars);
};
//--- Wenn Handle konnte nicht erstellt werden
if(handle==INVALID_HANDLE)
{
    //--- darüber schreiben und Nummer des Fehlers anzeigen
    PrintFormat("Handle des Indikators iChaikin für das Paar %s/%s konnte nicht erstellt werden",
        name,
        EnumToString(period),
        GetLastError());
    //--- Arbeit des Indikators ist früher geendet
    return(INIT_FAILED);
};
//--- Das Paar, Symbol/Zeitraumen, auf deren Chaikin Oscillator berechnet war, zeigen
short_name=StringFormat("iChaikin(%s/%s, %d, %d, %s, %s)",name,EnumToString(period),
    fast_ma_period,slow_ma_period,
    EnumToString(ma_method),EnumToString(applied_volume));
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- normale Initialisierung des Indikators
return(INIT_SUCCEEDED);
};

```

```

//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
    //--- Anzahl der Werte des Indikators iChaikin zu kopieren
    int values_to_copy;
    //--- Anzahl der berechneten Werte im Indikator finden
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() hat %d zurückgegeben, Fehlercode ist %d",calculated,GetLastError());
        return(0);
    }
    //--- wenn dies der erste Start der Berechnung des Indikators ist oder die Anzahl der
    //--- oder wenn Sie brauchen den Indikator für zwei oder mehr Balken (was bedeutet, dass
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
    {
        //--- wenn Array iChaikinBuffer größer als die Anzahl der Werte in iChaikin auf
        //--- sonst kopieren wir weniger als Größe der Indikator-Puffer
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    }
    else
    {
        //--- dann ist unser Indikator nicht das erste Mal berechnet, und seit dem letzten
        //--- nicht mehr als ein Balken für die Berechnung hinzugefügt war
        values_to_copy=(rates_total-prev_calculated)+1;
    }
    //--- Das Array iChaikinBuffer mit den Werten aus dem Indikator Chaikin Oscillator ausfüllen
    //--- Wenn FillArrayFromBuffer false zurückgegeben hat, dann die Daten nicht bereit stellen
    if(!FillArrayFromBuffer(iChaikinBuffer,handle,values_to_copy)) return(0);
    //--- Nachricht bilden
    string comm=StringFormat("%s ==> Aktualisierte Werte im Indikator %s: %d",
                             TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                             short_name,
                             values_to_copy);
    //--- Hilfsnachrichtung auf dem Chart anzeigen
    Comment(comm);
    //--- die Anzahl der Werte im Indikator Chaikin Oscillator speichern

```

```

    bars_calculated=calculated;
//--- den Wert prev_calculated für den nächsten Anruf zurückgeben
    return(rates_total);
};
//+-----+
//| Indikator-Puffer aus dem Indikator iChaikin ausfüllen |
//+-----+
bool FillArrayFromBuffer(double &values[], // Indikator-Puffer der Werte von Chaikin
                        int ind_handle,    // Handle von iChaikin
                        int amount        // Anzahl der Werte, die kopiert werden
                        )
{
//--- Fehlercode rücksetzen
    ResetLastError();
//--- Teil des Arrays iChaikinBuffer mit Werten auf Indikator-Puffer mit Index 0 ausfüllen
    if(CopyBuffer(ind_handle,0,0,amount,values)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iChaikin kopiert werden, Fehlercode: %d", GetLastError());
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden konnte
        return(false);
    }
//--- Alles gelang
    return(true);
};
//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
//--- Das Chart nach der Löschung des Indikators leeren
    Comment("");
};

```

## iCustom

Gibt das Handle des angegebenen Benutzerindikators zurück.

```
int iCustom(
    string      symbol,      // Symbolname
    ENUM_TIMEFRAMES period,  // Periode
    string      name         // Ordner/Name_des Benutzerindikators
    ...         // Liste der Eingabeparameter des Indikators
);
```

### Parameter

*symbol*

[in] Symbolname des Instrumentes, dessen Daten für Berechnung des Indikators verwendet werden. [NULL](#) bedeutet das laufende Symbol.

*period*

[in] Wert der Periode kann einer der Aufzählungswerte [ENUM\\_TIMEFRAMES](#) sein, 0 bedeutet das laufende Timeframe.

*name*

[in] Name des Benutzerindikators mit dem Pfad in Bezug auf Ordnerverzeichnis der Indikatoren (MQL5/Indicators/). Wenn ein Indikator in einem Unterverzeichnis liegt, zB. in MQL5/Indicators/Examples, muss der Name dementsprechend aussehen: "Examples\\Name\_des Indikators" (es ist notwendig, double-slash statt single slash als Begrenzer zu verwenden).

...

[in] [input-Parameter](#) eines Benutzerindikators, die durch Kommas getrennt sind. Typ und Parameterfolge müssen entsprechen. Wenn Parameter nicht angegeben werden, werden [Default-Werte](#) verwendet.

### Rückgabewert

Gibt das Handle des angegebenen technischen Indikators zurück, beim Misserfolg gibt [INVALID\\_HANDLE](#) zurück.

### Bemerkung

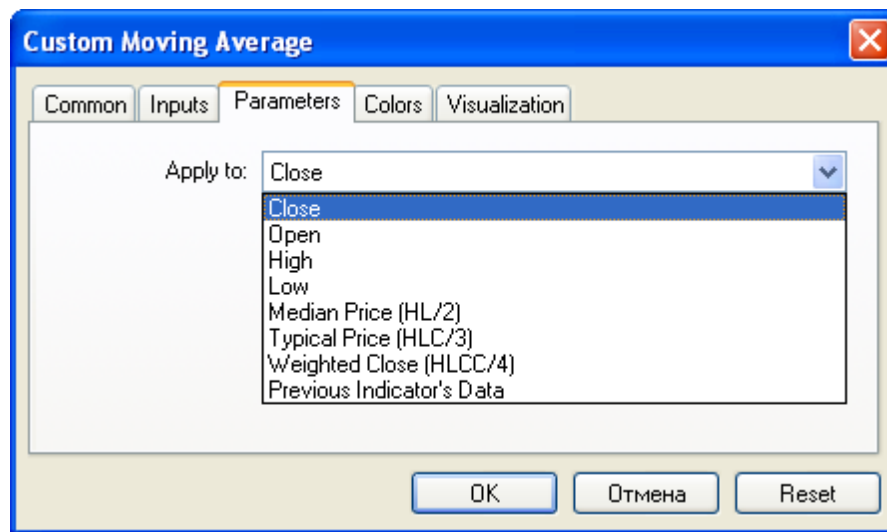
Benutzeranzeiger muss kompiliert werden (Datei mit Verbreitung EX5) und sich im Verzeichnis *MQL5/Indicators* des Kundenterminals oder in einem eingebetteten Verzeichnis befinden.

Die für Testenerforderliche Indikatoren werden automatisch aus Aufruf der Funktionen `iCustom()` bestimmt, wenn der entsprechende Parameter von der [Konstantzeile](#) vorgegeben ist. Für andere Fälle (Verwendung der Funktion [IndicatorCreate\(\)](#) oder Verwendung einer nicht Konstantzeile im Parameter, der den Namen des Indikators vorgibt) ist diese Eigenschaft [#property tester\\_indicator](#) erforderlich :

```
#property tester_indicator "indicator_name.ex5"
```

Wenn die [erste Aufrufform](#) im Indikator verwendet wird, kann beim Ablauf des Benutzerindikators im Registerblatt "Parameters" Daten für seine Berechnung angegeben werden. Wenn der Parameter

"Apply to" explizit nicht ausgewählt ist, wird die Berechnung mit den Werten "Close" durchgeführt.



Beim Aufruf eines Benutzerindikators aus dem mql5-Programm kann Parameter Applied\_Price oder handle eines anderen Parameters am letzten nach aller vom Benutzerindikator vorausgesehenen Eingabevariablen übertragen werden.

#### Sehen Sie auch

[Programmeigenschaften](#), [Zugang zu Zeitreihen und Indikatoren](#), [IndicatorCreate\(\)](#), [IndicatorRelease\(\)](#)

#### Beispiel:

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots 1
//---- plot Label1
#property indicator_label1 "Label1"
#property indicator_type1 DRAW_LINE
#property indicator_color1 clrRed
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//--- input parameters
input int MA_Period=21;
input int MA_Shift=0;
input ENUM_MA_METHOD MA_Method=MODE_SMA;
//--- indicator buffers
double Label1Buffer[];
//--- handle des Benutzerindikators Custom Moving Average.mq5
int MA_handle;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- indicator buffers mapping
```

```

SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
ResetLastError();
MA_handle=iCustom(NULL,0,"Examples\\Custom Moving Average",
    MA_Period,
    MA_Shift,
    MA_Method,
    PRICE_CLOSE // Schlusspreise verwenden
);
Print("MA_handle = ",MA_handle," error = ",GetLastError());
//---
return(INIT_SUCCEEDED);
;}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
    const int prev_calculated,
    const datetime &time[],
    const double &open[],
    const double &high[],
    const double &low[],
    const double &close[],
    const long &tick_volume[],
    const long &volume[],
    const int &spread[])
{
//--- Werte des Indikators Custom Moving Average in den Indikator-Puffer kopieren
int copy=CopyBuffer(MA_handle,0,0,rates_total,Label1Buffer);
Print("copy = ",copy," rates_total = ",rates_total);
//--- beim erfolglosen Versuch geben wir das aus
if(copy<=0)
    Print("Erfolgloser Versuch, Indikatorwerte Custom Moving Average zu erhalten");
//--- return value of prev_calculated for next call
return(rates_total);
;}
//+-----+

```

## iDEMA

Gibt das Handle des Indikators Double Exponential Moving Average zurück. Hat nur einen Puffer.

```
int iDEMA(
    string          symbol,          // Symbolname
    ENUM_TIMEFRAMES period,          // Periode
    int             ma_period,        // Mittelungsperiode
    int             ma_shift,         // horizontale Verschiebung des Indikators
    ENUM_APPLIED_PRICE applied_price // Preistyp oder Handle
);
```

### Parameter

*symbol*

[in] Symbolname des Instrumentes, dessen Daten für Berechnung des Indikators verwendet werden. [NULL](#) bedeutet das laufende Symbol.

*period*

[in] Wert der Periode kann einer der Aufzählungswerte [ENUM\\_TIMEFRAMES](#) sein, 0 bedeutet das laufende Timeframe.

*ma\_period*

[in] Periode(Anzahl der Bars) für Berechnung des Indikators.

*ma\_shift*

[in] Verschiebung des Indikators in Bezug auf das Preischart.

*applied\_price*

[in] der verwendete Preis. Kann eine der Preiskonstanten [ENUM\\_APPLIED\\_PRICE](#) oder handle eines anderen Indikators sein.

### Rückgabewert

Gibt das Handle des angegebenen technischen Indikators zurück, beim Misserfolg gibt [INVALID\\_HANDLE](#) zurück. Um Computerspeicher aus nicht genutzten Indikatoren zu befreien, wird die Funktion [IndicatorRelease\(\)](#) verwendet, zu deren Indikatorhandle passiert wird.

### Beispiel:

```
//+-----+
//|                                     Demo_iDEMA.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Der Indikator zeigt, wie Daten erhalten werden sollen"
#property description "Indikator-Puffer für den technischen Indikator iDEMA."
#property description "Symbol und Zeitrahmen, in denen der Indikator berechnet wird,"
#property description "sind durch die Parameter symbol und period angegeben."
```



```

#property description "Erstellungsweise des Handles ist durch den 'type' Parameter (Ty

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots 1
//--- iDEMA bauen
#property indicator_label1 "iDEMA"
#property indicator_type1 DRAW_LINE
#property indicator_color1 clrRed
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//+-----+
//| Aufzählung der Erstellungsweisen des Handles |
//+-----+
enum Creation
{
    Call_iDEMA,          // iDEMA verwenden
    Call_IndicatorCreate // IndicatorCreate verwenden
};
//--- Eingabeparameter
input Creation      type=Call_iDEMA;          // Funktionstyp
input int           ma_period=14;             // MA Periode
input int           ma_shift=0;               // Verschiebung
input ENUM_APPLIED_PRICE applied_price=PRICE_CLOSE; // Preistyp
input string        symbol=" ";               // Symbol
input ENUM_TIMEFRAMES period=PERIOD_CURRENT; // Zeitrahmen
//--- Indicator-Puffer
double iDEMABuffer[];
//--- Eine Variable um Handle des Indikators iDEMA zu speichern
int handle;
//--- Variable für Speicherung
string name=symbol;
//--- Name des Indikators auf dem Chart
string short_name;
//--- die Anzahl der Werte im Indikator Double Exponential Moving Average gespeichert
int bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- Bindung von Array zum Indikator-Puffer
    SetIndexBuffer(0,iDEMABuffer,INDICATOR_DATA);
    //--- Verschiebung eingeben
    PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
    //--- das Symbol, auf das der Indikator gebaut wird, definieren
    name=symbol;
    //--- Leerzeichen aus dem linken und rechten löschen
    StringTrimRight(name);

```

```

StringTrimLeft(name);
//--- Wenn nach diesem die Länge des String name ist Null
if(StringLen(name)==0)
{
    //--- das Symbol aus dem Chart, auf dem der Indikator läuft, nehmen
    name=_Symbol;
};
//--- Erstellen ein Handel des Indikators
if(type==Call_iDEMA)
    handle=iDEMA(name,period,ma_period,ma_shift,applied_price);
else
{
    //--- die Struktur mit Werten von Indikatorparametern ausfüllen
    MqlParam pars[3];
    //--- MA Periode
    pars[0].type=TYPE_INT;
    pars[0].integer_value=ma_period;
    //--- Verschiebung
    pars[1].type=TYPE_INT;
    pars[1].integer_value=ma_shift;
    //--- Preistyp
    pars[2].type=TYPE_INT;
    pars[2].integer_value=applied_price;
    handle=IndicatorCreate(name,period,IND_DEMA,3,pars);
};
//--- Wenn Handle konnte nicht erstellt werden
if(handle==INVALID_HANDLE)
{
    //--- darüber schreiben und Nummer des Fehlers anzeigen
    PrintFormat("Handle des Indikators iDEMA für das Paar %s/%s konnte nicht erstellt
                name,
                EnumToString(period),
                GetLastError());
    //--- Arbeit des Indikators ist früher geendet
    return(INIT_FAILED);
};
//--- Das Paar, Symbol/Zeitraumen, auf deren Double Exponential Moving Average berechne
short_name=StringFormat("iDEMA(%s/%s, %d, %d, %s)",name,EnumToString(period),
                        ma_period,ma_shift,EnumToString(applied_price));
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- normale Initialisierung des Indikators
return(INIT_SUCCEEDED);
};
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],

```

```

        const double &open[],
        const double &high[],
        const double &low[],
        const double &close[],
        const long &tick_volume[],
        const long &volume[],
        const int &spread[])
    {
        //--- Anzahl der Werte des Indikators iDEMA zu kopieren
        int values_to_copy;
        //--- Anzahl der berechneten Werte im Indikator finden
        int calculated=BarsCalculated(handle);
        if(calculated<=0)
        {
            PrintFormat("BarsCalculated() hat %d zurückgegeben, Fehlercode ist %d",calculate
            return(0);
        }
        ;}
        //--- wenn dies der erste Start der Berechnung des Indikators ist oder die Anzahl der
        //--- oder wenn Sie brauchen den Indikator für zwei oder mehr Balken (was bedeutet, da
        if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
        {
            //--- wenn Array iDEMABuffer größer als die Anzahl der Werte in iDEMA auf symbol
            //--- sonst kopieren wir weniger als Größe der Indikator-Puffer
            if(calculated>rates_total) values_to_copy=rates_total;
            else
                values_to_copy=calculated;
        }
        ;}
        else
        {
            //--- dann ist unser Indikator nicht das erste Mal berechnet, und seit dem letzten
            //--- nicht mehr als ein Balken für die Berechnung hinzugefügt war
            values_to_copy=(rates_total-prev_calculated)+1;
        }
        ;}
        //--- Das Array iDEMABuffer mit den Werten aus dem Indikator Double Exponential Moving
        //--- Wenn FillArrayFromBuffer false zurückgegeben hat, dann die Daten nicht bereit sind
        if(!FillArrayFromBuffer(iDEMABuffer,ma_shift,handle,values_to_copy)) return(0);
        //--- Nachricht bilden
        string comm=StringFormat("%s ==> Aktualisierte Werte im Indikator %s: %d",
                                TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                                short_name,
                                values_to_copy);
        //--- Hilfsnachrichtung auf dem Chart anzeigen
        Comment(comm);
        //--- die Anzahl der Werte im Indikator Double Exponential Moving Average speichern
        bars_calculated=calculated;
        //--- den Wert prev_calculated für den nächsten Anruf zurückgeben
        return(rates_total);
    }
    ;}
    //+-----+
    //| Indikator-Puffer aus dem Indikator iDEMA ausfüllen |

```

```

//+-----+
bool FillArrayFromBuffer(double &values[], // Indikator-Puffer der Werte von Double E
                        int shift,        // Verschiebung
                        int ind_handle,    // Handle von iDEMA
                        int amount        // Anzahl der Werte, die kopiert werden
                        )
{
//--- Fehlercode rücksetzen
    ResetLastError();
//--- Teil des Arrays iDEMABuffer mit Werten auf Indikator-Puffer mit Index 0 ausfüllen
    if(CopyBuffer(ind_handle,0,-shift,amount,values)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iDEMA kopiert werden, Fehlercode: %d", GetLastError());
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden konnte
        return(false);
    }
//--- Alles gelang
    return(true);
};
//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
//--- Das Chart nach der Löschung des Indikators leeren
    Comment("");
};

```

## iDeMarker

Gibt das Handle des Indikators DeMarker zurück. Hat nur einen Puffer.

```
int iDeMarker(
    string      symbol,      // Symbolname
    ENUM_TIMEFRAMES period,  // Periode
    int         ma_period    // Mittelungswert
);
```

### Parameter

*symbol*

[in] Symbolname des Instrumentes, dessen Daten für Berechnung des Indikators verwendet werden. NULL bedeutet das laufende Symbol.

*period*

[in] Wert kann einer der Aufzählungswerte ENUM\_TIMEFRAMES sein, 0 bedeutet das laufende Timeframe.

*ma\_period*

[in] Mittelungsperiode für Berechnung des Indikators.

### Rückgabewert

Gibt das Handle des angegebenen technischen Indikators zurück, beim Misserfolg gibt INVALID\_HANDLE zurück. Um Computerspeicher aus nicht genutzten Indikatoren zu befreien, wird die Funktion IndicatorRelease() verwendet, zu deren Indikatorhandle passiert wird.

### Beispiel:

```
//+-----+
//|                                     Demo_iDeMarker.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Der Indikator zeigt, wie Daten erhalten werden sollen"
#property description "Indikator-Puffer für den technischen Indikator iDeMarker."
#property description "Symbol und Zeitrahmen, in denen der Indikator berechnet wird,"
#property description "sind durch die Parameter symbol und period angegeben."
#property description "Erstellungsweise des Handles ist durch den 'type' Parameter (T)"

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1

//--- iDeMarker bauen
#property indicator_label1  "iDeMarker"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
```

```

#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- Horizontale Ebenen im Indikatorsfenster
#property indicator_level1  0.3
#property indicator_level2  0.7
//+-----+
//| Aufzählung der Erstellungsweisen des Handles |
//+-----+
enum Creation
{
    Call_iDeMarker,          // iDeMarker verwenden
    Call_IndicatorCreate     // IndicatorCreate verwenden
};
//--- Eingabeparameter
input Creation             type=Call_iDeMarker;      // Funktionstyp
input int                  ma_period=14;             // MA Periode
input string               symbol=" ";               // Symbol
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;    // Zeitrahmen
//--- Indicator-Puffer
double                    iDeMarkerBuffer[];
//--- Eine Variable um Handle des Indikators iDeMarker zu speichern
int    handle;
//--- Variable für Speicherung
string name=symbol;
//--- Name des Indikators auf dem Chart
string short_name;
//--- die Anzahl der Werte im Indikator DeMarker gespeichert wird
int    bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- Bindung von Array zum Indikator-Puffer
    SetIndexBuffer(0,iDeMarkerBuffer,INDICATOR_DATA);
    //--- das Symbol, auf das der Indikator gebaut wird, definieren
    name=symbol;
    //--- Leerzeichen aus dem linken und rechten löschen
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- Wenn nach diesem die Länge des String name ist Null
    if(StringLen(name)==0)
    {
        //--- das Symbol aus dem Chart, auf dem der Indikator läuft, nehmen
        name=_Symbol;
    };
    //--- Erstellen ein Handel des Indikators
    if(type==Call_iDeMarker)
        handle=iDeMarker(name,period,ma_period);
}

```

```

else
{
    //--- die Struktur mit Werten von Indikatorparametern ausfüllen
    MqlParam pars[1];
    //--- MA Periode
    pars[0].type=TYPE_INT;
    pars[0].integer_value=ma_period;
    handle=IndicatorCreate(name,period,IND_DEMARKER,1,pars);
};

//--- Wenn Handle konnte nicht erstellt werden
if(handle==INVALID_HANDLE)
{
    //--- darüber schreiben und Nummer des Fehlers anzeigen
    PrintFormat("Handle des Indikators iDeMarker für das Paar %s/%s konnte nicht erst
                name,
                EnumToString(period),
                GetLastError());

    //--- Arbeit des Indikators ist früher geendet
    return(INIT_FAILED);
};

//--- Das Paar von Symbol/Zeitraumen, auf deren DeMarker berechnet war, anzeigen
short_name=StringFormat("iDeMarker(%s/%s, period=%d)",name,EnumToString(period),ma_
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- normale Initialisierung des Indikators
return(INIT_SUCCEEDED);
};

//+-----+
//| Custom indicator iteration function |
//+-----+

int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
    //--- Anzahl der Werte des Indikators iDeMarker zu kopieren
    int values_to_copy;
    //--- Anzahl der berechneten Werte im Indikator finden
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() hat %d zurückgegeben, Fehlercode ist %d",calculated,
        return(0);
    };
};

```

```

//--- wenn dies der erste Start der Berechnung des Indikators ist oder die Anzahl der
//--- oder wenn Sie brauchen den Indikator für zwei oder mehr Balken (was bedeutet, da
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
    {
        //--- wenn Array iDeMarkerBuffer größer als die Anzahl der Werte in iDeMarker au
        //--- sonst kopieren wir weniger als Größe der Indikator-Puffer
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    };
else
{
    //--- dann ist unser Indikator nicht das erste Mal berechnet, und seit dem letzte
    //--- nicht mehr als ein Balken für die Berechnung hinzugefügt war
    values_to_copy=(rates_total-prev_calculated)+1;
};

//--- Das Array iDeMarkerBuffer mit den Werten aus dem Indikator DeMarker ausfüllen
//--- Wenn FillArrayFromBuffer false zurückgegeben hat, dann die Daten nicht bereit s
    if(!FillArrayFromBuffer(iDeMarkerBuffer,handle,values_to_copy)) return(0);
//--- Nachricht bilden
    string comm=StringFormat("%s ==> Aktualisierte Werte im Indikator %s: %d",
                                TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                                short_name,
                                values_to_copy);

//--- Hilfsnachrichtung auf dem Chart anzeigen
    Comment(comm);

//--- die Anzahl der Werte im Indikator DeMarker speichern
    bars_calculated=calculated;

//--- den Wert prev_calculated für den nächsten Anruf zurückgeben
    return(rates_total);
};

//+-----+
//| Indikator-Puffer aus dem Indikator iDeMarker ausfüllen |
//+-----+
bool FillArrayFromBuffer(double &values[], // Indikator-Puffer der Werte von DeMarker
                        int ind_handle, // Handle von iDeMarker
                        int amount // Anzahl der Werte, die kopiert werden
                        )
{
    //--- Fehlercode rücksetzen
    ResetLastError();

    //--- Teil des Arrays iDeMarkerBuffer mit Werten auf Indikator-Puffer mit Index 0 ausf
    if(CopyBuffer(ind_handle,0,0,amount,values)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iDeMarker kopiert werden, Fehl
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berech
        return(false);
    };
//--- Alles gelang

```



```
    return(true);  
};  
//+-----+  
//| Indicator deinitialization function |  
//+-----+  
void OnDeinit(const int reason)  
{  
    //--- Das Chart nach der Löschung des Indikators leeren  
    Comment("");  
};
```

## iEnvelopes

Gibt das Handle des Indikators Envelopes zurück.

```
int iEnvelopes(
    string          symbol,          // Symbolname
    ENUM_TIMEFRAMES period,          // Periode
    int             ma_period,        // Periode für Berechnung der Mittellingslinie
    int             ma_shift,         // horizontale Verschiebung des Indikators
    ENUM_MA_METHOD  ma_method,        // Mittelungstyp
    ENUM_APPLIED_PRICE applied_price, // Preistyp oder handle
    double          deviation         // Abweichung der Grenzen von der Mittellinie
);
```

### Parameter

*symbol*

[in] Symbolname des Instrumentes, dessen Daten für Berechnung des Indikators verwendet werden. [NULL](#) bedeutet das laufende Symbol.

*period*

[in] Wert der Periode kann einer der Aufzählungswerte [ENUM\\_TIMEFRAMES](#), sein 0 bedeutet das laufende Timeframe.

*ma\_period*

[in] Mittelungsperiode der Hauptlinie des Indikators.

*ma\_shift*

[in] Verschiebung des Indikators in Bezug auf das Preischart.

*ma\_method*

[in] Glättungsmethode. Kann einer der Aufzählungswerte [ENUM\\_MA\\_METHOD](#) sein.

*applied\_price*

[in] Der verwendete Preis. Kann eine der Preiskonstanten [ENUM\\_APPLIED\\_PRICE](#) oder Handle eines anderen Indikators sein.

*deviation*

[in] Abweichung von der Grundlinie in Prozenten.

### Rückgabewert

Gibt das Handle des angegebenen technischen Indikators zurück, beim Misserfolg gibt [INVALID\\_HANDLE](#) zurück. Um Computerspeicher aus nicht genutzten Indikatoren zu befreien, wird die Funktion [IndicatorRelease\(\)](#) verwendet, zu deren Indikatorhandle passiert wird.

### Bemerkung

Puffernummern: 0 - UPPER\_LINE, 1 - LOWER\_LINE.

### Beispiel:

```
//+-----+
```

```

//|                                     Demo_iEnvelopes.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Der Indikator zeigt, wie Daten erhalten werden sollen"
#property description "Indikator-Puffer für den technischen Indikator iEnvelopes."
#property description "Symbol und Zeitrahmen, in denen der Indikator berechnet wird,"
#property description "sind durch die Parameter symbol und period angegeben."
#property description "Erstellungsweise des Handles ist durch den 'type' Parameter (Ty

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   2
//--- Upper bauen
#property indicator_label1  "Upper"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- Lower bauen
#property indicator_label2  "Lower"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//+-----+
//| Aufzählung der Erstellungsweisen des Handles |
//+-----+
enum Creation
{
    Call_iEnvelopes,      // iEnvelopes verwenden
    Call_IndicatorCreate   // IndicatorCreate verwenden
};
//--- Eingabeparameter
input Creation          type=Call_iEnvelopes;      // Funktionstyp
input int               ma_period=14;              // MA Periode
input int               ma_shift=0;                // Verschiebung
input ENUM_MA_METHOD    ma_method=MODE_SMA;        // Art der Glättung
input ENUM_APPLIED_PRICE applied_price=PRICE_CLOSE; // Preistyp
input double            deviation=0.1;             // Abweichung von Grenzen aus M
input string            symbol=" ";               // Symbol
input ENUM_TIMEFRAMES   period=PERIOD_CURRENT;    // Zeitrahmen
//--- Indikator-Puffer
double      UpperBuffer[];
double      LowerBuffer[];
//--- Eine Variable um Handle des Indikators iEnvelopes zu speichern

```

```

int    handle;
//--- Variable für Speicherung
string name=symbol;
//--- Name des Indikators auf dem Chart
string short_name;
//--- die Anzahl der Werte im Indikator Envelopes gespeichert wird
int    bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- Bindung von Arrays zu den Indikator-Puffern
    SetIndexBuffer(0,UpperBuffer,INDICATOR_DATA);
    SetIndexBuffer(1,LowerBuffer,INDICATOR_DATA);
//--- Verschiebung jede Linie definieren
    PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
    PlotIndexSetInteger(1,PLOT_SHIFT,ma_shift);
//--- das Symbol, auf das der Indikator gebaut wird, definieren
    name=symbol;
//--- Leerzeichen aus dem linken und rechten löschen
    StringTrimRight(name);
    StringTrimLeft(name);
//--- Wenn nach diesem die Länge des String name ist Null
    if(StringLen(name)==0)
    {
        //--- das Symbol aus dem Chart, auf dem der Indikator läuft, nehmen
        name=_Symbol;
    }
//--- Erstellen ein Handel des Indikators
    if(type==Call_iEnvelopes)
        handle=iEnvelopes(name,period,ma_period,ma_shift,ma_method,applied_price,deviat:
    else
    {
        //--- die Struktur mit Werten von Indikatorparametern ausfüllen
        MqlParam pars[5];
        //--- MA Periode
        pars[0].type=TYPE_INT;
        pars[0].integer_value=ma_period;
        //--- Verschiebung
        pars[1].type=TYPE_INT;
        pars[1].integer_value=ma_shift;
        //--- Art der Glättung
        pars[2].type=TYPE_INT;
        pars[2].integer_value=ma_method;
        //--- Preistyp
        pars[3].type=TYPE_INT;
        pars[3].integer_value=applied_price;
        //--- Preistyp

```

```

        pars[4].type=TYPE_DOUBLE;
        pars[4].double_value=deviation;
        handle=IndicatorCreate(name,period,IND_ENVELOPES,5,pars);
    };
//--- Wenn Handle konnte nicht erstellt werden
    if(handle==INVALID_HANDLE)
    {
        //--- darüber schreiben und Nummer des Fehlers anzeigen
        PrintFormat("Handle des Indikators iEnvelopes für das Paar %s/%s konnte nicht e
            name,
            EnumToString(period),
            GetLastError());
        //--- Arbeit des Indikators ist früher geendet
        return(INIT_FAILED);
    };
//--- Das Paar von Symbol/Zeitraumen, auf deren Envelopes berechnet war, anzeigen
    short_name=StringFormat("iEnvelopes(%s/%s, %d, %d, %s,%s, %G)",name,EnumToString(pe
    ma_period,ma_shift,EnumToString(ma_method),EnumToString(applied_price),deviation);
    IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- normale Initialisierung des Indikators
    return(INIT_SUCCEEDED);
};
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{
//--- Anzahl der Werte des Indikators iEnvelopes zu kopieren
    int values_to_copy;
//--- Anzahl der berechneten Werte im Indikator finden
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() hat %d zurückgegeben, Fehlercode ist %d",calculated
        return(0);
    };
//--- wenn dies der erste Start der Berechnung des Indikators ist oder die Anzahl der
//--- oder wenn Sie brauchen den Indikator für zwei oder mehr Balken (was bedeutet, da
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
    {

```

```

    //--- wenn Array UpperBuffer größer als die Anzahl der Werte in iEnvelopes auf s
    //--- sonst kopieren wir weniger als Größe der Indikator-Puffer
    if(calculated>rates_total) values_to_copy=rates_total;
    else
        values_to_copy=calculated;
    ;}
else
{
    //--- dann ist unser Indikator nicht das erste Mal berechnet, und seit dem letzte
    //--- nicht mehr als ein Balken für die Berechnung hinzugefügt war
    values_to_copy=(rates_total-prev_calculated)+1;
    ;}
//--- Arrays UpperBuffer und LowerBuffer mit den Werten aus dem Indikator Envelopes au
//--- Wenn FillArrayFromBuffer false zurückgegeben hat, dann die Daten nicht bereit s
    if(!FillArraysFromBuffers(UpperBuffer,LowerBuffer,ma_shift,handle,values_to_copy))
//--- Nachricht bilden
    string comm=StringFormat("%s ==> Aktualisierte Werte im Indikator %s: %d",
                                TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                                short_name,
                                values_to_copy);
//--- Hilfsnachrichtung auf dem Chart anzeigen
    Comment(comm);
//--- die Anzahl der Werte im Indikator Envelopes speichern
    bars_calculated=calculated;
//--- den Wert prev_calculated für den nächsten Anruf zurückgeben
    return(rates_total);
    ;}
//+-----+
//| Indikator-Puffer aus dem Indikator iEnvelopesator ausfüllen |
//+-----+
bool FillArraysFromBuffers(double &upper_values[], // Indikator-Puffer der Obergrenze
                           double &lower_values[], // Indikator-Puffer der Untergrenze
                           int shift, // Verschiebung
                           int ind_handle, // Handle des Indikators iEnvelopes
                           int amount // Anzahl der Werte, die kopiert werden
                           )
{
    //--- Fehlercode rücksetzen
    ResetLastError();
    //--- Teil des Arrays UpperBuffer mit Werten auf Indikator-Puffer mit Index 0 ausfüllen
    if(CopyBuffer(ind_handle,0,-shift,amount,upper_values)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iEnvelopes kopiert werden, Fehlercode: %d",
                    GetLastError());
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet wurde
        return(false);
    }
    ;}
//--- Teil des Arrays LowerBuffer mit Werten auf Indikator-Puffer mit Index 1 ausfüllen
    if(CopyBuffer(ind_handle,1,-shift,amount,lower_values)<0)
    {

```

```
//--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
PrintFormat("Daten konnte nicht aus dem Indikator iEnvelopes kopiert werden, Fehlercode: %d", GetLastError());
//--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet wurde
return(false);
};
//--- Alles gelang
return(true);
};
//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
//--- Das Chart nach der Löschung des Indikators leeren
Comment("");
};
```

## iForce

Gibt das Handle des Indikators Force Index zurück. Hat nur einen Puffer.

```
int iForce(
    string          symbol,          // Symbolname
    ENUM_TIMEFRAMES period,          // Periode
    int             ma_period,        // Mittelungsperiode
    ENUM_MA_METHOD   ma_method,       // Glaettungstyp
    ENUM_APPLIED_VOLUME applied_volume // Volumentyp für Berechnung
);
```

### Parameter

*symbol*

[in] Symbolname des Instrumentes, dessen Daten für Berechnung des Indikators verwendet werden. [NULL](#) bedeutet das laufende Symbol.

*period*

[in] Wert der Periode kann einer der Aufzählungswerte [ENUM\\_TIMEFRAMES](#) sein, 0 bedeutet das laufende Timeframe.

*ma\_period*

[in] Mittelungsperiode für Berechnung des Indikators.

*ma\_method*

[in] Glaettungsmethode. Kann einer der Aufzählungswerte [ENUM\\_MA\\_METHOD](#) sein.

*applied\_volume*

[in] Das verwendete Volumen. Kann einer der Aufzählungswerte [ENUM\\_APPLIED\\_VOLUME](#) sein.

### Rueckgabewert

Gibt das Handle des angegebenen technischen Indikators zurück, beim Misserfolg gibt [INVALID\\_HANDLE](#) zurück. Um Computerspeicher aus nicht genutzten Indikatoren zu befreien, wird die Funktion [IndicatorRelease\(\)](#) verwendet, zu deren Indikatorhandle passiert wird.

### Beispiel:

```
//+-----+
//|                                     Demo_iForce.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Der Indikator zeigt, wie Daten erhalten werden sollen"
#property description "Indikator-Puffer für den technischen Indikator iForce."
#property description "Symbol und Zeitrahmen, in denen der Indikator berechnet wird,"
#property description "sind durch die Parameter symbol und period angegeben."
#property description "Erstellungsweise des Handles ist durch den 'type' Parameter (T)
```



```

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots 1
//--- iForce bauen
#property indicator_label1 "iForce"
#property indicator_type1 DRAW_LINE
#property indicator_color1 clrLightSeaGreen
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//+-----+
//| Aufzählung der Erstellungsweisen des Handles |
//+-----+
enum Creation
{
    Call_iForce,          // iForce verwenden
    Call_IndicatorCreate  // IndicatorCreate verwenden
};
//--- Eingabeparameter
input Creation          type=Call_iForce;          // Funktionstyp
input int               ma_period=13;              // MA Periode
input ENUM_MA_METHOD    ma_method=MODE_SMA;        // Art der Glättung
input ENUM_APPLIED_VOLUME applied_volume=VOLUME_TICK; // Volumentyp
input string            symbol=" ";                // Symbol
input ENUM_TIMEFRAMES   period=PERIOD_CURRENT;     // Zeitrahmen
//--- Indicator-Puffer
double iForceBuffer[];
//--- Eine Variable um Handle des Indikators Art der Glättung zu speichern
int handle;
//--- Variable für Speicherung
string name=symbol;
//--- Name des Indikators auf dem Chart
string short_name;
//--- die Anzahl der Werte im Indikator Force gespeichert wird
int bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- Bindung von Array zum Indikator-Puffer
    SetIndexBuffer(0,iForceBuffer,INDICATOR_DATA);
    //--- das Symbol, auf das der Indikator gebaut wird, definieren
    name=symbol;
    //--- Leerzeichen aus dem linken und rechten löschen
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- Wenn nach diesem die Länge des String name ist Null
    if(StringLen(name)==0)

```

```

    {
        //--- das Symbol aus dem Chart, auf dem der Indikator läuft, nehmen
        name=_Symbol;
    };
//--- Erstellen ein Handel des Indikators
if(type==Call_iForce)
    handle=iForce(name,period,ma_period,ma_method,applied_volume);
else
    {
        //--- die Struktur mit Werten von Indikatorparametern ausfüllen
        MqlParam pars[3];
        //--- MA Periode
        pars[0].type=TYPE_INT;
        pars[0].integer_value=ma_period;
        //--- Art der Glättung
        pars[1].type=TYPE_INT;
        pars[1].integer_value=ma_method;
        //--- Volumentyp
        pars[2].type=TYPE_INT;
        pars[2].integer_value=applied_volume;
        //--- Preistyp
        handle=IndicatorCreate(name,period,IND_FORCE,3,pars);
    };
//--- Wenn Handle konnte nicht erstellt werden
if(handle==INVALID_HANDLE)
    {
        //--- darüber schreiben und Nummer des Fehlers anzeigen
        PrintFormat("Handle des Indikators iForce für das Paar %s/%s konnte nicht erstellt werden",
                    name,
                    EnumToString(period),
                    GetLastError());
        //--- Arbeit des Indikators ist früher geendet
        return(INIT_FAILED);
    };
//--- Das Paar von Symbol/Zeitraumen, auf deren Force berechnet war, anzeigen
short_name=StringFormat("iForce(%s/%s, %d, %s, %s)",name,EnumToString(period),
                        ma_period,EnumToString(ma_method),EnumToString(applied_volume));
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- normale Initialisierung des Indikators
return(INIT_SUCCEEDED);
};
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],

```

```

        const double &low[],
        const double &close[],
        const long &tick_volume[],
        const long &volume[],
        const int &spread[])

{
//--- Anzahl der Werte des Indikators iForce zu kopieren
    int values_to_copy;
//--- Anzahl der berechneten Werte im Indikator finden
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() hat %d zurückgegeben, Fehlercode ist %d",calculated,GetLastError());
        return(0);
    }
//--- wenn dies der erste Start der Berechnung des Indikators ist oder die Anzahl der
//--- oder wenn Sie brauchen den Indikator für zwei oder mehr Balken (was bedeutet, dass
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
    {
        //--- wenn Array iForceBuffer größer als die Anzahl der Werte in iForce auf symbol
        //--- sonst kopieren wir weniger als Größe der Indikator-Puffer
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    }
else
    {
        //--- dann ist unser Indikator nicht das erste Mal berechnet, und seit dem letzten
        //--- nicht mehr als ein Balken für die Berechnung hinzugefügt war
        values_to_copy=(rates_total-prev_calculated)+1;
    }
//--- Das Array iForceBuffer mit den Werten aus dem Indikator Force ausfüllen
//--- Wenn FillArrayFromBuffer false zurückgegeben hat, dann die Daten nicht bereit sind
    if(!FillArrayFromBuffer(iForceBuffer,handle,values_to_copy)) return(0);
//--- Nachricht bilden
    string comm=StringFormat("%s ==> Aktualisierte Werte im Indikator %s: %d",
                               TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                               short_name,
                               values_to_copy);
//--- Hilfsnachrichtung auf dem Chart anzeigen
    Comment(comm);
//--- die Anzahl der Werte im Indikator Force speichern
    bars_calculated=calculated;
//--- den Wert prev_calculated für den nächsten Anruf zurückgeben
    return(rates_total);
};

//+-----+
//| Indikator-Puffer aus dem Indikator iForce ausfüllen |
//+-----+
bool FillArrayFromBuffer(double &values[], // Indikator-Puffer der Werte von Force Ind

```

```

        int ind_handle,    // Handle des Indikators iForce
        int amount        // Anzahl der Werte, die kopiert werden
    )

    {
//--- Fehlercode rücksetzen
    ResetLastError();
//--- Teil des Arrays iForceBuffer mit Werten auf Indikator-Puffer mit Index 0 ausfüllen
    if(CopyBuffer(ind_handle,0,0,amount,values)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iForce kopiert werden, Fehlercode: %d", GetLastError());
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden konnte
        return(false);
    }
//--- Alles gelang
    return(true);
}

//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
//--- Das Chart nach der Löschung des Indikators leeren
    Comment("");
}

```

## iFractals

Gibt das Handle des Indikators Fractals zurück.

```
int iFractals(
    string      symbol,      // Symbolname
    ENUM_TIMEFRAMES period   // Periode
);
```

### Parameter

*symbol*

[in] Symbolname des Instrumentes, dessen Daten für Berechnung des Indikators verwendet werden. NULL bedeutet das laufende Symbol.

*period*

[in] Wert der Periode kann einer der Aufzählungswerte ENUM\_TIMEFRAMES sein, 0 bedeutet das laufende Timeframe.

### Rückgabewert

Gibt das Handle des angegebenen technischen Indikators zurück, beim Misserfolg gibt INVALID\_HANDLE zurück. Um Computerspeicher aus nicht genutzten Indikatoren zu befreien, wird die Funktion IndicatorRelease() verwendet, zu deren Indikatorhandle passiert wird.

### Bemerkung

Puffernummern: 0 - UPPER\_LINE, 1 - LOWER\_LINE.

### Beispiel:

```
//+-----+
//|                                     Demo_iFractals.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Der Indikator zeigt, wie Daten erhalten werden sollen"
#property description "Indikator-Puffer für den technischen Indikator iFractals."
#property description "Symbol und Zeitrahmen, in denen der Indikator berechnet wird,"
#property description "sind durch die Parameter symbol und period angegeben."
#property description "Erstellungsweise des Handles ist durch den 'type' Parameter (T)"

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots 1
#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots 2

//--- FractalUp bauen
```

```

#property indicator_label1 "FractalUp"
#property indicator_type1 DRAW_ARROW
#property indicator_color1 clrBlue
//--- FractalDown bauen
#property indicator_label2 "FractalDown"
#property indicator_type2 DRAW_ARROW
#property indicator_color2 clrRed
//+-----+
//| Aufzählung der Erstellungsweisen des Handles |
//+-----+
enum Creation
{
    Call_iFractals,          // iFractals verwenden
    Call_IndicatorCreate     // IndicatorCreate verwenden
};
//--- Eingabeparameter
input Creation              type=Call_iFractals;          // Funktionstyp
input string                symbol=" ";                  // Symbol
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;       // Zeitrahmen
//--- Indicator-Puffer
double      FractalUpBuffer[];
double      FractalDownBuffer[];
//--- Eine Variable um Handle des Indikators iFractals zu speichern
int         handle;
//--- Variable für Speicherung
string name=symbol;
//--- Name des Indikators auf dem Chart
string short_name;
//--- die Anzahl der Werte im Indikator Fractals gespeichert wird
int         bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- Bindung von Arrays zu den Indikator-Puffern
    SetIndexBuffer(0,FractalUpBuffer,INDICATOR_DATA);
    SetIndexBuffer(1,FractalDownBuffer,INDICATOR_DATA);
    //--- Codes der Symbole aus Reihe von Wingdings für PLOT_ARROW Eigenschaften definieren
    PlotIndexSetInteger(0,PLOT_ARROW,217); // Pfeil nach oben
    PlotIndexSetInteger(1,PLOT_ARROW,218); // Pfeil nach unten
    //--- das Symbol, auf das der Indikator gebaut wird, definieren
    name=symbol;
    //--- Leerzeichen aus dem linken und rechten löschen
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- Wenn nach diesem die Länge des String name ist Null
    if(StringLen(name)==0)
    {

```

```

    //--- das Symbol aus dem Chart, auf dem der Indikator läuft, nehmen
    name=_Symbol;
    ;}

//--- Erstellen ein Handel des Indikators
if(type==Call_iFractals)
    handle=iFractals(name,period);
else
    handle=IndicatorCreate(name,period,IND_FRACTALS);
//--- Wenn Handle konnte nicht erstellt werden
if(handle==INVALID_HANDLE)
{
    //--- darüber schreiben und Nummer des Fehlers anzeigen
    PrintFormat("Handle des Indikators iFractals für das Paar %s/%s konnte nicht erst
                name,
                EnumToString(period),
                GetLastError());
    //--- Arbeit des Indikators ist früher geendet
    return(INIT_FAILED);
};

//--- Das Paar von Symbol/Zeitraumen, auf deren Fractals berechnet war, anzeigen
short_name=StringFormat("iFractals(%s/%s)",name,EnumToString(period));
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- normale Initialisierung des Indikators
return(INIT_SUCCEEDED);
};

//+-----+
//| Custom indicator iteration function |
//+-----+

int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{
    //--- Anzahl der Werte des Indikators iFractals zu kopieren
    int values_to_copy;
    //--- Anzahl der berechneten Werte im Indikator finden
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() hat %d zurückgegeben, Fehlercode ist %d",calculated,GetLastError());
        return(0);
    };
    //--- wenn dies der erste Start der Berechnung des Indikators ist oder die Anzahl der

```

```

//--- oder wenn Sie brauchen den Indikator für zwei oder mehr Balken (was bedeutet, da
if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
{
    //--- wenn Array FractalUpBuffer größer als die Anzahl der Werte in iFractals au
    //--- sonst kopieren wir weniger als Größe der Indikator-Puffer
    if(calculated>rates_total) values_to_copy=rates_total;
    else
        values_to_copy=calculated;
};
else
{
    //--- dann ist unser Indikator nicht das erste Mal berechnet, und seit dem letzte
    //--- nicht mehr als ein Balken für die Berechnung hinzugefügt war
    values_to_copy=(rates_total-prev_calculated)+1;
};

//--- Arrays FractalUpBuffer und FractalDownBuffer mit den Werten aus dem Indikator Fr
//--- Wenn FillArrayFromBuffer false zurückgegeben hat, dann die Daten nicht bereit si
if(!FillArraysFromBuffers(FractalUpBuffer,FractalDownBuffer,handle,values_to_copy))
//--- Nachricht bilden
string comm=StringFormat("%s ==> Aktualisierte Werte im Indikator %s: %d",
                        TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                        short_name,
                        values_to_copy);

//--- Hilfsnachrichtung auf dem Chart anzeigen
Comment(comm);

//--- die Anzahl der Werte im Indikator Fractals speichern
bars_calculated=calculated;

//--- den Wert prev_calculated für den nächsten Anruf zurückgeben
return(rates_total);
};

//+-----+
//| Indikator-Puffer aus dem Indikator iFractals ausfüllen |
//+-----+

bool FillArraysFromBuffers(double &up_arrows[], // Indikator-Puffer der Pfeile
                        double &down_arrows[], // Indikator-Puffer der Pfeile
                        int ind_handle, // Handle des Indikators iFract
                        int amount // Anzahl der Werte, die kopier

)

{
    //--- Fehlercode rücksetzen
    ResetLastError();

    //--- Teil des Arrays FractalUpBuffer mit Werten auf Indikator-Puffer mit Index 0 aus
    if(CopyBuffer(ind_handle,0,0,amount,up_arrows)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iFractals ins Array FractalUp
                        GetLastError());

        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berech
        return(false);
    };
};

```



```

//--- Teil des Arrays FractalDownBuffer mit Werten auf Indikator-Puffer mit Index 1 au
    if(CopyBuffer(ind_handle,1,0,amount,down_arrows)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iFractals ins Array FractalDov
                        GetLastError());
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berech
        return(false);
    }
//--- Alles gelang
    return(true);
};
//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
    //--- Das Chart nach der Löschung des Indikators leeren
    Comment("");
};

```

## iFrAMA

Gibt das Handle des Indikators Fractal Adaptive Moving Average zurück. Hat nur einen Puffer.

```
int iFrAMA(
    string          symbol,          // Symbolname
    ENUM_TIMEFRAMES period,          // Periode
    int             ma_period,        // Mittelungsperiode
    int             ma_shift,         // horizontale Verschiebung des Indikators
    ENUM_APPLIED_PRICE applied_price // Preistyp oder Handle
);
```

### Parameter

*symbol*

[in] Symbolname des Instrumentes, dessen Daten für Berechnung des Indikators verwendet werden. [NULL](#) bedeutet das laufende Symbol.

*period*

[in] Wert kann eine der Aufzählungswerte [ENUM\\_TIMEFRAMES](#) sein, 0 bedeutet das laufende Timeframe.

*ma\_period*

[in] Periode(Anzahl der Bars) für Berechnung des Indikators.

*ma\_shift*

[in] Verschiebung des Indikators in Bezug auf das Preischart.

*applied\_price*

[in] Der verwendete Preis. Kann einer der Preiskonstanten [ENUM\\_APPLIED\\_PRICE](#) oder Handle eines anderen Indikators sein.

### Rückgabewert

Gibt das Handle des angegebenen technischen Indikators zurück, beim Misserfolg gibt [INVALID\\_HANDLE](#) zurück. Um Computerspeicher aus nicht genutzten Indikatoren zu befreien, wird die Funktion [IndicatorRelease\(\)](#) verwendet, zu deren Indikatorhandle passiert wird.

### Beispiel:

```
//+-----+
//|                                     Demo_iFrAMA.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Der Indikator zeigt, wie Daten erhalten werden sollen"
#property description "Indikator-Puffer für den technischen Indikator iFrAMA."
#property description "Symbol und Zeitrahmen, in denen der Indikator berechnet wird,"
#property description "sind durch die Parameter symbol und period angegeben."
```

```

#property description "Erstellungsweise des Handles ist durch den 'type' Parameter (Ty

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots 1
//--- iFrAMA bauen
#property indicator_label1 "iFrAMA"
#property indicator_type1 DRAW_LINE
#property indicator_color1 clrBlue
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//+-----+
//| Aufzählung der Erstellungsweisen des Handles |
//+-----+
enum Creation
{
    Call_iFrAMA,          // iFrAMA verwenden
    Call_IndicatorCreate  // IndicatorCreate verwenden
};
//--- Eingabeparameter
input Creation          type=Call_iFrAMA;          // Funktionstyp
input int               ma_period=14;              // MA Periode
input int               ma_shift=0;                // Verschiebung
input ENUM_APPLIED_PRICE applied_price=PRICE_CLOSE; // Preistyp
input string            symbol=" ";                // Symbol
input ENUM_TIMEFRAMES   period=PERIOD_CURRENT;     // Zeitrahmen
//--- Indicator-Puffer
double iFrAMABuffer[];
//--- Eine Variable um Handle des Indikators iFrAMA zu speichern
int handle;
//--- Variable für Speicherung
string name=symbol;
//--- Name des Indikators auf dem Chart
string short_name;
//--- die Anzahl der Werte im Indikator Fractal Adaptive Moving Average gespeichert w
int bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- Bindung von Array zum Indikator-Puffer
    SetIndexBuffer(0,iFrAMABuffer,INDICATOR_DATA);
    //--- Verschiebung definieren
    PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
    //--- das Symbol, auf das der Indikator gebaut wird, definieren
    name=symbol;
    //--- Leerzeichen aus dem linken und rechten löschen
    StringTrimRight(name);

```

```

StringTrimLeft(name);
//--- Wenn nach diesem die Länge des String name ist Null
if(StringLen(name)==0)
{
    //--- das Symbol aus dem Chart, auf dem der Indikator läuft, nehmen
    name=_Symbol;
};
//--- Erstellen ein Handel des Indikators
if(type==Call_iFrAMA)
    handle=iFrAMA(name,period,ma_period,ma_shift,applied_price);
else
{
    //--- die Struktur mit Werten von Indikatorparametern ausfüllen
    MqlParam pars[3];
    //--- MA Periode
    pars[0].type=TYPE_INT;
    pars[0].integer_value=ma_period;
    //--- Verschiebung
    pars[1].type=TYPE_INT;
    pars[1].integer_value=ma_shift;
    //--- Preistyp
    pars[2].type=TYPE_INT;
    pars[2].integer_value=applied_price;
    //--- Preistyp
    handle=IndicatorCreate(name,period,IND_FRAMA,3,pars);
};
//--- Wenn Handle konnte nicht erstellt werden
if(handle==INVALID_HANDLE)
{
    //--- darüber schreiben und Nummer des Fehlers anzeigen
    PrintFormat("Handle des Indikators iFrAMA für das Paar %s/%s konnte nicht erstellt werden",
        name,
        EnumToString(period),
        GetLastError());
    //--- Arbeit des Indikators ist früher geendet
    return(INIT_FAILED);
};
//--- Das Paar von Symbol/Zeitraumen, auf deren iFrAMA berechnet war, anzeigen
short_name=StringFormat("iFrAMA(%s/%s, %d, %d, %s)",name,EnumToString(period),
    ma_period,ma_shift,EnumToString(applied_price));
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- normale Initialisierung des Indikators
return(INIT_SUCCEEDED);
};
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
    const int prev_calculated,

```

```

        const datetime &time[],
        const double &open[],
        const double &high[],
        const double &low[],
        const double &close[],
        const long &tick_volume[],
        const long &volume[],
        const int &spread[])
    {
        //--- Anzahl der Werte des Indikators iFrAMA zu kopieren
        int values_to_copy;
        //--- Anzahl der berechneten Werte im Indikator finden
        int calculated=BarsCalculated(handle);
        if(calculated<=0)
        {
            PrintFormat("BarsCalculated() hat %d zurückgegeben, Fehlercode ist %d",calculated,GetLastError());
            return(0);
        }
        //--- wenn dies der erste Start der Berechnung des Indikators ist oder die Anzahl der
        //--- oder wenn Sie brauchen den Indikator für zwei oder mehr Balken (was bedeutet, dass
        if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
        {
            //--- wenn Array iFrAMABuffer größer als die Anzahl der Werte in iFrAMA auf symbol
            //--- sonst kopieren wir weniger als Größe der Indikator-Puffer
            if(calculated>rates_total) values_to_copy=rates_total;
            else
                values_to_copy=calculated;
        }
        else
        {
            //--- dann ist unser Indikator nicht das erste Mal berechnet, und seit dem letzten
            //--- nicht mehr als ein Balken für die Berechnung hinzugefügt war
            values_to_copy=(rates_total-prev_calculated)+1;
        }
        //--- Das Array iFrAMABuffer mit den Werten aus dem Indikator Fractal Adaptive Moving Average
        //--- Wenn FillArrayFromBuffer false zurückgegeben hat, dann die Daten nicht bereit sind
        if(!FillArrayFromBuffer(iFrAMABuffer,ma_shift,handle,values_to_copy)) return(0);
        //--- Nachricht bilden
        string comm=StringFormat("%s ==> Aktualisierte Werte im Indikator %s: %d",
                                   TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                                   short_name,
                                   values_to_copy);
        //--- Hilfsnachrichtung auf dem Chart anzeigen
        Comment(comm);
        //--- die Anzahl der Werte im Indikator Fractal Adaptive Moving Average speichern
        bars_calculated=calculated;
        //--- den Wert prev_calculated für den nächsten Anruf zurückgeben
        return(rates_total);
    }
    //+-----+

```

```

//| Indikator-Puffer aus dem Indikator iFrAMA ausfüllen |
//+-----+
bool FillArrayFromBuffer(double &values[], // Indikator-Puffer der Werte von Fractal
                        int shift,        // Verschiebung
                        int ind_handle,    // Handle des Indikators iFrAMA
                        int amount        // Anzahl der Werte, die kopiert werden
                        )
{
    //--- Fehlercode rücksetzen
    ResetLastError();
    //--- Teil des Arrays iFrAMABuffer mit Werten auf Indikator-Puffer mit Index 0 ausfüllen
    if(CopyBuffer(ind_handle,0,-shift,amount,values)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iFrAMA kopiert werden, Fehlercode: %d", GetLastError());
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden konnte
        return(false);
    }
    //--- Alles gelang
    return(true);
}
//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
    //--- Das Chart nach der Löschung des Indikators leeren
    Comment("");
}

```

## iGator

Gibt das Handle des Indikators Gator zurück. Oscillator zeigt den Unterschied zwischen der blauen und der roten Linien des Alligators (oberer Chart) und Unterschied zwischen der roten und der grünen Linien (unterer Chart).

```
int iGator(
    string          symbol,          // Symbolname
    ENUM_TIMEFRAMES period,          // Periode
    int             jaw_period,      // Periode für Berechnung der Kiefer
    int             jaw_shift,       // horizontale Verschiebung der Kiefer
    int             teeth_period,    // Periode für Berechnung der Zähne
    int             teeth_shift,     // horizontale Verschiebung der Kiefer
    int             lips_period,     // Periode für Berechnung der Lippen
    int             lips_shift,      // horizontale Verschiebung der Lippen
    ENUM_MA_METHOD  ma_method,       // Glättungstyp
    ENUM_APPLIED_PRICE applied_price // Preistyp oder Handle
);
```

### Parameter

*symbol*

[in] Symbolname des Instrumentes, dessen Daten für Berechnung des Indikators verwendet werden. [NULL](#) bedeutet das laufende Symbol.

*period*

[in] Wert der Periode kann einer der Aufzählungswerte [ENUM\\_TIMEFRAMES](#) sein, 0 bedeutet das laufende Timeframe.

*jaw\_period*

[in] Mittelungsperiode der blauen Linie (Kiefer des Alligators).

*jaw\_shift*

[in] Verschiebung der blauen Linie des Alligators in Bezug auf das Preischart. Nicht direkt verbunden mit der visuellen Verschiebung des Histogramms des Indikators.

*teeth\_period*

[in] Mittelungsperiode der roten Linie (Zähne des Alligators).

*teeth\_shift*

[in] Verschiebung der roten Linie des Alligators in Bezug auf das Preischart. Nicht direkt verbunden mit der visuellen Verschiebung des Histogramms des Indikators.

*lips\_period*

[in] Mittelungsperiode der grünen Linie (Lippen des Alligators).

*lips\_shift*

[in] Verschiebung der grünen Linie des Alligators in Bezug auf das Preischart. Nicht direkt verbunden mit der visuellen Verschiebung des Histogramms des Indikators.

*ma\_method*

[in] Mittelungsmethode. Kann einer der Werte [ENUM\\_MA\\_METHOD](#) sein.

*applied\_price*

[in] Der verwendete Preis. Kann eine der Preiskonstanten [ENUM\\_APPLIED\\_PRICE](#) oder Handle eines anderen Indikators sein.

### Rückgabewert

Gibt das Handle des angegebenen technischen Indikators zurück, beim Misserfolg gibt [INVALID\\_HANDLE](#) zurück. Um Computerspeicher aus nicht genutzten Indikatoren zu befreien, wird die Funktion [IndicatorRelease\(\)](#) verwendet, zu deren Indikatorhandle passiert wird.

### Bemerkung

Puffernummern: 0 - UPPER\_HISTOGRAM, 1- Farbenpuffer des oberen Histogramms, 2 - LOWER\_HISTOGRAM, 3- Farbenpuffer des unteren Histogramms.

### Beispiel:

```
//+-----+
//|                                     Demo_iGator.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Der Indikator zeigt, wie Daten erhalten werden sollen"
#property description "Indikator-Puffer für den technischen Indikator iGator."
#property description "Symbol und Zeitrahmen, in denen der Indikator berechnet wird,"
#property description "sind durch die Parameter symbol und period angegeben."
#property description "Erstellungsweise des Handles ist durch den 'type' Parameter (Ty
#property description "Alle andere Parameter sind wie im normalen Indikator Gator Osc

#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots 2
//--- GatorUp bauen
#property indicator_label1 "GatorUp"
#property indicator_type1  DRAW_COLOR_HISTOGRAM
#property indicator_color1 clrGreen, clrRed
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//--- GatorDown bauen
#property indicator_label2 "GatorDown"
#property indicator_type2  DRAW_COLOR_HISTOGRAM
#property indicator_color2 clrGreen, clrRed
#property indicator_style2 STYLE_SOLID
#property indicator_width2 1
//+-----+
//| Aufzählung der Erstellungsweisen des Handles |
//+-----+

enum Creation
```



```

{
    Call_iGator,          // iGator verwenden
    Call_IndicatorCreate  // IndicatorCreate verwenden
};
//--- Eingabeparameter
input Creation           type=Call_iGator;      // Funktionstyp
input string             symbol=" ";            // Symbol
input ENUM_TIMEFRAMES    period=PERIOD_CURRENT; // Zeitrahmen
input int                jaw_period=13;         // Zeitraum für die Kiefer-Linie
input int                jaw_shift=8;           // Verschiebung der Kiefer-Linie
input int                teeth_period=8;        // Zeitraum für die Zähne-Linie
input int                teeth_shift=5;         // Verschiebung der Zähne-Linie
input int                lips_period=5;         // Zeitraum für die Lippen-Linie
input int                lips_shift=3;          // Verschiebung der Lippen-Linie
input ENUM_MA_METHOD      MA_method=MODE_SMMMA; // Methode der Mittelung der Alligatoren
input ENUM_APPLIED_PRICE  applied_price=PRICE_MEDIAN; // Typ der Preise aus der der Alligatoren
//--- Indikator-Puffer
double      GatorUpBuffer[];
double      GatorUpColors[];
double      GatorDownBuffer[];
double      GatorDownColors[];
//--- Eine Variable um Handle des Indikators iGator zu speichern
int         handle;
//--- Variable für Speicherung
string name=symbol;
//--- Name des Indikators auf dem Chart
string short_name;
//--- Verschiebung für die obere und untere Histogramme
int shift;
//--- die Anzahl der Werte im Indikator Gator Oscillator gespeichert wird
int bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- Bindung von Arrays zu den Indikator-Puffern
    SetIndexBuffer(0,GatorUpBuffer,INDICATOR_DATA);
    SetIndexBuffer(1,GatorUpColors,INDICATOR_COLOR_INDEX);
    SetIndexBuffer(2,GatorDownBuffer,INDICATOR_DATA);
    SetIndexBuffer(3,GatorDownColors,INDICATOR_COLOR_INDEX);
/*
    Alle Verschiebungen, die in den Parametern angegeben werden, gehören zum Indikator Z
    Daher produzieren diese Verschiebungen keine Verschiebung des Indikators Gator, und
    auf dessen Werte die Indikatorwerte von Gator Oscillator konstruiert sind!
*/
//--- Berechnung der Verschiebung für den oberen und unteren Histogramm, die die Abwei
    shift=MathMin(jaw_shift,teeth_shift);
    PlotIndexSetInteger(0,PLOT_SHIFT,shift);

```

```

//--- Trotz der Tatsache, dass der Indikator zwei Histogramme hat, die gleiche Verschiebung haben
PlotIndexSetInteger(1,PLOT_SHIFT,shift);

//--- das Symbol, auf das der Indikator gebaut wird, definieren
name=symbol;
//--- Leerzeichen aus dem linken und rechten löschen
StringTrimRight(name);
StringTrimLeft(name);
//--- Wenn nach diesem die Länge des String name ist Null
if(StringLen(name)==0)
{
    //--- das Symbol aus dem Chart, auf dem der Indikator läuft, nehmen
    name=_Symbol;
};
//--- Erstellen ein Handle des Indikators
if(type==Call_iGator)
    handle=iGator(name,period,jaw_period,jaw_shift,teeth_period,teeth_shift,
        lips_period,lips_shift,MA_method,applied_price);
else
{
    //--- die Struktur mit Werten von Indikatorparametern ausfüllen
    MqlParam pars[8];
    //--- Perioden und Verschiebungen der Alligator-Linien
    pars[0].type=TYPE_INT;
    pars[0].integer_value=jaw_period;
    pars[1].type=TYPE_INT;
    pars[1].integer_value=jaw_shift;
    pars[2].type=TYPE_INT;
    pars[2].integer_value=teeth_period;
    pars[3].type=TYPE_INT;
    pars[3].integer_value=teeth_shift;
    pars[4].type=TYPE_INT;
    pars[4].integer_value=lips_period;
    pars[5].type=TYPE_INT;
    pars[5].integer_value=lips_shift;
    //--- Art der Glättung
    pars[6].type=TYPE_INT;
    pars[6].integer_value=MA_method;
    //--- Preistyp
    pars[7].type=TYPE_INT;
    pars[7].integer_value=applied_price;
    //--- Handle erstellen
    handle=IndicatorCreate(name,period,IND_GATOR,8,pars);
};
//--- Wenn Handle konnte nicht erstellt werden
if(handle==INVALID_HANDLE)
{
    //--- darüber schreiben und Nummer des Fehlers anzeigen
    PrintFormat("Handle des Indikators iGator für das Paar %s/%s konnte nicht erstellt werden");
}

```

```

        name,
        EnumToString(period),
        GetLastError());

    //--- Arbeit des Indikators ist früher geendet
    return(INIT_FAILED);

; }

//--- Das Paar, Symbol/Zeitraumen, auf deren Gator Oscillator berechnet war, zeigen
short_name=StringFormat("iGator(%s/%s, %d, %d, %d, %d, %d, %d)", name, EnumToString(p
        jaw_period, jaw_shift, teeth_period, teeth_shift, lips_period, 1
    IndicatorSetString(INDICATOR_SHORTNAME, short_name);
//--- normale Initialisierung des Indikators
return(INIT_SUCCEEDED);

; }

//+-----+
//| Custom indicator iteration function |
//+-----+

int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])

{
    //--- Anzahl der Werte des Indikators iGator zu kopieren
    int values_to_copy;
    //--- Anzahl der berechneten Werte im Indikator finden
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() hat %d zurückgegeben, Fehlercode ist %d", calculated, GetLastError());
        return(0);
    }
; }

//--- wenn dies der erste Start der Berechnung des Indikators ist oder die Anzahl der
//--- oder wenn Sie brauchen den Indikator für zwei oder mehr Balken (was bedeutet, da
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
    {
        //--- wenn Array GatorUpBuffer größer als die Anzahl der Werte in iGator auf sym
        //--- sonst kopieren wir weniger als Größe der Indikator-Puffer
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    }
; }
else
{
    //--- dann ist unser Indikator nicht das erste Mal berechnet, und seit dem letzten
    //--- nicht mehr als ein Balken für die Berechnung hinzugefügt war

```

```

        values_to_copy=(rates_total-prev_calculated)+1;
    };
//--- Die Arrays mit den Werten aus dem Indikator Gator Oscillator ausfüllen
//--- Wenn FillArraysFromBuffer false zurückgegeben hat, dann die Daten nicht bereit s
    if(!FillArraysFromBuffers(GatorUpBuffer,GatorUpColors,GatorDownBuffer,GatorDownColo
        shift,handle,values_to_copy)) return(0);
//--- Nachricht bilden
    string comm=StringFormat("%s ==> Aktualisierte Werte im Indikator %s: %d",
        TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
        short_name,
        values_to_copy);
//--- Hilfsnachrichtung auf dem Chart anzeigen
    Comment(comm);
//--- die Anzahl der Werte im Indikator Gator Oscillator speichern
    bars_calculated=calculated;
//--- den Wert prev_calculated für den nächsten Anruf zurückgeben
    return(rates_total);
};
//+-----+
//| Indikator-Puffer aus dem Indikator iGator ausfüllen |
//+-----+
bool FillArraysFromBuffers(double &ups_buffer[], // Indikator-Puffer für die o
    double &up_color_buffer[], // Indikator-Puffer für die I
    double &downs_buffer[], // Indikator-Puffer für die u
    double &downs_color_buffer[], // Indikator-Puffer für die I
    int u_shift, // Verschiebung für die obere
    int ind_handle, // Handle des Indikators iGat
    int amount // Anzahl der Werte, die kop
)
{
//--- Fehlercode rücksetzen
    ResetLastError();
//--- Teil des Arrays GatorUpBuffer mit Werten auf Indikator-Puffer mit Index 0 ausfü
    if(CopyBuffer(ind_handle,0,-u_shift,amount,ups_buffer)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iGator kopiert werden, Fehlerc
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berech
        return(false);
    };
//--- Teil des Arrays GatorUpColors mit Werten auf Indikator-Puffer mit Index 1 ausfü
    if(CopyBuffer(ind_handle,1,-u_shift,amount,up_color_buffer)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iGator kopiert werden, Fehlerc
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berech
        return(false);
    };
};

```

```

//--- Teil des Arrays GatorDownBuffer mit Werten auf Indikator-Puffer mit Index 2 aus
if(CopyBuffer(ind_handle,2,-u_shift,amount,downs_buffer)<0)
{
    //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
    PrintFormat("Daten konnte nicht aus dem Indikator iGator kopiert werden, Fehlercode: %d", GetLastError());
    //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden konnte
    return(false);
};

//--- Teil des Arrays GatorDownColors mit Werten auf Indikator-Puffer mit Index 3 aus
if(CopyBuffer(ind_handle,3,-u_shift,amount,downs_color_buffer)<0)
{
    //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
    PrintFormat("Daten konnte nicht aus dem Indikator iGator kopiert werden, Fehlercode: %d", GetLastError());
    //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden konnte
    return(false);
};

//--- Alles gelang
return(true);
};

//+-----+
//| Indicator deinitialization function |
//+-----+

void OnDeinit(const int reason)
{
    //--- Das Chart nach der Löschung des Indikators leeren
    Comment("");
};

```

## ilchimoku

Gibt das Handle des Indikators Ichimoku Kinko Hyo zurück.

```
int iIchimoku(
    string          symbol,           // Symbolname
    ENUM_TIMEFRAMES period,          // Periode
    int             tenkan_sen,       // Periode Tenkan-sen
    int             kijun_sen,        // Periode Kijun-sen
    int             senkou_span_b     // Periode Senkou Span B
);
```

### Parameter

*symbol*

[in] Symbolname des Instrumentes, dessen Daten für Berechnung des Indikators verwendet werden. [NULL](#) das laufende Symbol.

*period*

[in] Wert der Periode kann einer der Aufzählungswerte [ENUM\\_TIMEFRAMES](#) sein, 0 bedeutet das laufende Timeframe.

*tenkan\_sen*

[in] Mittelungsperiode Tenkan Sen.

*kijun\_sen*

[in] Mittelungsperiode Kijun Sen.

*senkou\_span\_b*

[in] Mittelungsperiode Senkou Span B.

### Rückgabewert

Gibt das Handle des angegebenen technischen Indikators zurück, beim Misserfolg gibt [INVALID\\_HANDLE](#) zurück. Um Computerspeicher aus nicht genutzten Indikatoren zu befreien, wird die Funktion [IndicatorRelease\(\)](#) verwendet, zu deren Indikatorhandle passiert wird.

### Bemerkung

Puffernummern: 0 - TENKANSEN\_LINE, 1 - KIJUNSEN\_LINE, 2 - SENKOUSPANA\_LINE, 3 - SENKOUSPANB\_LINE, 4 - CHINKOSPAN\_LINE.

### Beispiel:

```
//+-----+
//|                                     Demo_iIchimoku.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Der Indikator zeigt, wie Daten erhalten werden sollen"
```

```

#property description "Indikator-Puffer für den technischen Indikator iIchimoku."
#property description "Symbol und Zeitrahmen, in denen der Indikator berechnet wird,"
#property description "sind durch die Parameter symbol und period angegeben."
#property description "Erstellungsweise des Handles ist durch den 'type' Parameter (Ty
#property description "Alle andere Parameter sind wie im normalen Ichimoku Kinko Hyo."

#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots 4
//--- Tenkan_sen konstruieren
#property indicator_label1 "Tenkan_sen"
#property indicator_type1 DRAW_LINE
#property indicator_color1 clrRed
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//--- Kijun_sen konstruieren
#property indicator_label2 "Kijun_sen"
#property indicator_type2 DRAW_LINE
#property indicator_color2 clrBlue
#property indicator_style2 STYLE_SOLID
#property indicator_width2 1
//--- Senkou_Span konstruieren
#property indicator_label3 "Senkou Span A;Senkou Span B" // Zwei Felder werden in Dat
#property indicator_type3 DRAW_FILLING
#property indicator_color3 clrSandyBrown, clrThistle
#property indicator_style3 STYLE_SOLID
#property indicator_width3 1
//--- Chinkou_Span konstruieren
#property indicator_label4 "Chinkou_Span"
#property indicator_type4 DRAW_LINE
#property indicator_color4 clrLime
#property indicator_style4 STYLE_SOLID
#property indicator_width4 1
//+-----+
//| Aufzählung der Erstellungsweisen des Handles |
//+-----+
enum Creation
{
    Call_iIchimoku,          // iIchimoku verwenden
    Call_IndicatorCreate     // IndicatorCreate verwenden
};
//--- Eingabeparameter
input Creation              type=Call_iIchimoku;          // Funktionstyp
input int                  tenkan_sen=9;                  // Periode von Tenkan-sen
input int                  kijun_sen=26;                  // Periode von Kijun-sen
input int                  senkou_span_b=52;              // Periode von Senkou Span B
input string                symbol=" ";                  // Symbol
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;        // Zeitrahmen
//--- Indikator-Puffer

```

```

double      Tenkan_sen_Buffer[];
double      Kijun_sen_Buffer[];
double      Senkou_Span_A_Buffer[];
double      Senkou_Span_B_Buffer[];
double      Chinkou_Span_Buffer[];
//--- Eine Variable um Handle des Indikators iIchimoku zu speichern
int         handle;
//--- Variable für Speicherung
string name=symbol;
//--- Name des Indikators auf dem Chart
string short_name;
//--- die Anzahl der Werte im Indikator Ichimoku Kinko Hyo gespeichert wird
int         bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- Bindung von Arrays zu den Indikator-Puffern
SetIndexBuffer(0,Tenkan_sen_Buffer,INDICATOR_DATA);
SetIndexBuffer(1,Kijun_sen_Buffer,INDICATOR_DATA);
SetIndexBuffer(2,Senkou_Span_A_Buffer,INDICATOR_DATA);
SetIndexBuffer(3,Senkou_Span_B_Buffer,INDICATOR_DATA);
SetIndexBuffer(4,Chinkou_Span_Buffer,INDICATOR_DATA);
//--- Die Verschiebung für Kanal Senkou Span auf kijun_sen Balken nach Zukunft definiert
PlotIndexSetInteger(2,PLOT_SHIFT,kijun_sen);
//--- Für Linie Chinkou Span, es ist nicht erforderlich, die Verschiebung anzugeben, c
//--- in iIchimoku mit Verschiebung gespeichert
//--- das Symbol, auf das der Indikator gebaut wird, definieren
name=symbol;
//--- Leerzeichen aus dem linken und rechten löschen
StringTrimRight(name);
StringTrimLeft(name);
//--- Wenn nach diesem die Länge des String name ist Null
if(StringLen(name)==0)
{
//--- das Symbol aus dem Chart, auf dem der Indikator läuft, nehmen
name=_Symbol;
; }
//--- Erstellen ein Handel des Indikators
if(type==Call_iIchimoku)
handle=iIchimoku(name,period,tenkan_sen,kijun_sen,senkou_span_b);
else
{
//--- die Struktur mit Werten von Indikatorparametern ausfüllen
MqlParam pars[3];
//--- Perioden und Verschiebungen der Alligator-Linien
pars[0].type=TYPE_INT;
pars[0].integer_value=tenkan_sen;

```



```

    pars[1].type=TYPE_INT;
    pars[1].integer_value=kijun_sen;
    pars[2].type=TYPE_INT;
    pars[2].integer_value=senkou_span_b;
    //--- Handle erstellen
    handle=IndicatorCreate(name,period,IND_ICHIMOKU,3,pars);

; }

//--- Wenn Handle konnte nicht erstellt werden
if(handle==INVALID_HANDLE)
{
    //--- darüber schreiben und Nummer des Fehlers anzeigen
    PrintFormat("Handle des Indikators iBWMFI für das Paar %s/%s konnte nicht erstel
                name,
                EnumToString(period),
                GetLastError());

    //--- Arbeit des Indikators ist früher geendet
    return(INIT_FAILED);

; }

//--- Das Paar, Symbol/Zeitraumen, auf deren Ichimoku Kinko Hyo berechnet war, zeigen
short_name=StringFormat("iIchimoku(%s/%s, %d, %d, %d)",name,EnumToString(period),
                        tenkan_sen,kijun_sen,senkou_span_b);

IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- normale Initialisierung des Indikators
return(INIT_SUCCEEDED);

; }

//+-----+
//| Custom indicator iteration function |
//+-----+

int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])

{
    //--- Anzahl der Werte des Indikators iIchimoku zu kopieren
    int values_to_copy;
    //--- Anzahl der berechneten Werte im Indikator finden
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() hat %d zurückgegeben, Fehlercode ist %d",calculated,
                    GetLastError());
        return(0);

    ; }

    //--- wenn dies der erste Start der Berechnung des Indikators ist oder die Anzahl der

```

```

//--- oder wenn Sie brauchen den Indikator für zwei oder mehr Balken (was bedeutet, da
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
    {
        //--- wenn Array Tenkan_sen_Buffer größer als die Anzahl der Werte in iIchimoku
        //--- sonst kopieren wir weniger als Größe der Indikator-Puffer
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    }
else
{
    //--- dann ist unser Indikator nicht das erste Mal berechnet, und seit dem letzten
    //--- nicht mehr als ein Balken für die Berechnung hinzugefügt war
    values_to_copy=(rates_total-prev_calculated)+1;
}

//--- Die Arrays mit den Werten aus dem Indikator Ichimoku Kinko Hyo ausfüllen
//--- Wenn FillArraysFromBuffer false zurückgegeben hat, dann die Daten nicht bereit s
    if(!FillArraysFromBuffers(Tenkan_sen_Buffer,Kijun_sen_Buffer,Senkou_Span_A_Buffer,Senkou_Span_B_Buffer,Chinkou_Span_Buffer,
        kijun_sen,handle,values_to_copy)) return(0);
//--- Nachricht bilden
    string comm=StringFormat("%s ==> Aktualisierte Werte im Indikator %s: %d",
        TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
        short_name,
        values_to_copy);
//--- Hilfsnachrichtung auf dem Chart anzeigen
    Comment(comm);
//--- die Anzahl der Werte im Indikator Ichimoku Kinko Hyo speichern
    bars_calculated=calculated;
//--- den Wert prev_calculated für den nächsten Anruf zurückgeben
    return(rates_total);
};

//+-----+
//| Indikator-Puffer aus dem Indikator iIchimoku ausfüllen |
//+-----+
bool FillArraysFromBuffers(double &tenkan_sen_buffer[], // Indikator-Puffer für die Tenkan-Sen
    double &kijun_sen_buffer[], // Indikator-Puffer für die Kijun-Sen
    double &senkou_span_A_buffer[], // Indikator-Puffer für die Senkou Span A
    double &senkou_span_B_buffer[], // Indikator-Puffer für die Senkou Span B
    double &chinkou_span_buffer[], // Indikator-Puffer für die Chinkou Span
    int senkou_span_shift, // Verschiebung der Linie
    int ind_handle, // Handle des Indikators
    int amount // Anzahl der Werte, die kopiert werden sollen
)
{
    //--- Fehlercode rücksetzen
    ResetLastError();
    //--- Teil des Arrays Tenkan_sen_Buffer mit Werten auf Indikator-Puffer mit Index 0 auf
    if(CopyBuffer(ind_handle,0,0,amount,tenkan_sen_buffer)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen

```

```

        PrintFormat("1.Daten konnte nicht aus dem Indikator iIchimoku kopiert werden, Fehlercode: %d", GetLastError());
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet wurde
        return(false);
    };
}

//--- Teil des Arrays Kijun_sen_Buffer mit Werten auf Indikator-Puffer mit Index 1 auslesen
if(CopyBuffer(ind_handle,1,0,amount,kijun_sen_buffer)<0)
{
    //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
    PrintFormat("2.Daten konnte nicht aus dem Indikator iIchimoku kopiert werden, Fehlercode: %d", GetLastError());
    //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet wurde
    return(false);
};

//--- Teil des Arrays Chinkou_Span_Buffer mit Werten auf Indikator-Puffer mit Index 2 auslesen
//--- Mit senkou_span_shift>0 Linie wird die Linie zu senkou_span_shift Balken nach Zukunft verschoben
if(CopyBuffer(ind_handle,2,-senkou_span_shift,amount,senkou_span_A_buffer)<0)
{
    //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
    PrintFormat("3.Daten konnte nicht aus dem Indikator iIchimoku kopiert werden, Fehlercode: %d", GetLastError());
    //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet wurde
    return(false);
};

//--- Teil des Arrays Senkou_Span_A_Buffer mit Werten auf Indikator-Puffer mit Index 3 auslesen
//--- Mit senkou_span_shift>0 Linie wird die Linie zu senkou_span_shift Balken nach Zukunft verschoben
if(CopyBuffer(ind_handle,3,-senkou_span_shift,amount,senkou_span_B_buffer)<0)
{
    //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
    PrintFormat("4.Daten konnte nicht aus dem Indikator iIchimoku kopiert werden, Fehlercode: %d", GetLastError());
    //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet wurde
    return(false);
};

//--- Teil des Arrays Senkou_Span_B_Buffer mit Werten auf Indikator-Puffer mit Index 4 auslesen
//--- Beim Kopieren der Linie Chinkou Span, es ist nicht erforderlich, die Verschiebung zu berücksichtigen
//--- in iIchimoku mit Verschiebung gespeichert
if(CopyBuffer(ind_handle,4,0,amount,chinkou_span_buffer)<0)
{
    //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
    PrintFormat("5.Daten konnte nicht aus dem Indikator iIchimoku kopiert werden, Fehlercode: %d", GetLastError());
    //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet wurde
    return(false);
};

//--- Alles gelang
return(true);
};

//+-----+
//| Indicator deinitialization function |

```

```
//+-----+  
void OnDeinit(const int reason)  
{  
    //--- Das Chart nach der Löschung des Indikators leeren  
    Comment("");  
};
```

## iBWMFI

Gibt das Handle des Indikators Market Facilitation Index zurück. Hat nur einen Puffer.

```
int iBWMFI(
    string          symbol,          // Symbolname
    ENUM_TIMEFRAMES period,          // Periode
    ENUM_APPLIED_VOLUME applied_volume // Volumentyp für Berechnung
);
```

### Parameter

*symbol*

[in] Symbolname des Instrumentes, dessen Daten für Berechnung des Indikators verwendet werden. NULL bedeutet das laufende Symbol.

*period*

[in] Wert der Periode kann einer der Aufzählungswerte ENUM\_TIMEFRAMES sein, 0 bedeutet das laufende Timeframe.

*applied\_volume*

[in] Das verwendete Volumen. Kann einer der Aufzählungswerte ENUM\_APPLIED\_VOLUME sein.

### Rückgabewert

Gibt das Handle des angegebenen technischen Indikators zurück, beim Misserfolg gibt INVALID\_HANDLE zurück. Um Computerspeicher aus nicht genutzten Indikatoren zu befreien, wird die Funktion IndicatorRelease() verwendet, zu deren Indikatorhandle passiert wird.

### Beispiel:

```
//+-----+
//|                                     Demo_iBWMFI.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Der Indikator zeigt, wie Daten erhalten werden sollen"
#property description "Indikator-Puffer für den technischen Indikator iBWMFI."
#property description "Symbol und Zeitrahmen, in denen der Indikator berechnet wird,"
#property description "sind durch die Parameter symbol und period angegeben."
#property description "Erstellungsweise des Handles ist durch den 'type' Parameter (T)"

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots 1
//--- iBWMFI bauen
#property indicator_label1 "iBWMFI"
#property indicator_type1  DRAW_COLOR_HISTOGRAM
#property indicator_color1 clrLime,clrSaddleBrown,clrBlue,clrPink
```

```

#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//+-----+
//| Aufzählung der Erstellungsweisen des Handles |
//+-----+
enum Creation
{
    Call_iBWMFI,          // iBWMFI verwenden
    Call_IndicatorCreate  // IndicatorCreate verwenden
};
//--- Eingabeparameter
input Creation          type=Call_iBWMFI;          // Funktionstyp
input ENUM_APPLIED_VOLUME applied_volume=VOLUME_TICK; // Volumentyp
input string            symbol=" ";                // Symbol
input ENUM_TIMEFRAMES   period=PERIOD_CURRENT;     // Zeitrahmen
//--- Indicator-Puffer
double      iBWMFIBuffer[];
double      iBWMFIColors[];
//--- Eine Variable um Handle des Indikators iBWMFI zu speichern
int         handle;
//--- Variable für Speicherung
string name=symbol;
//--- Name des Indikators auf dem Chart
string short_name;
//--- die Anzahl der Werte im Indikator Market Facilitation Index von Bill Williams ge
int         bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- Bindung von Arrays zu den Indikator-Puffern
    SetIndexBuffer(0,iBWMFIBuffer,INDICATOR_DATA);
    SetIndexBuffer(1,iBWMFIColors,INDICATOR_COLOR_INDEX);
    //--- das Symbol, auf das der Indikator gebaut wird, definieren
    name=symbol;
    //--- Leerzeichen aus dem linken und rechten löschen
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- Wenn nach diesem die Länge des String name ist Null
    if(StringLen(name)==0)
    {
        //--- das Symbol aus dem Chart, auf dem der Indikator läuft, nehmen
        name=_Symbol;
    };
    //--- Erstellen ein Handel des Indikators
    if(type==Call_iBWMFI)
        handle=iBWMFI(name,period,applied_volume);
    else

```

```

{
    //--- die Struktur mit Werten von Indikatorparametern ausfüllen
    MqlParam pars[1];
    //--- Volumentyp
    pars[0].type=TYPE_INT;
    pars[0].integer_value=applied_volume;
    handle=IndicatorCreate(name,period,IND_BWMFI,1,pars);
};
//--- Wenn Handle konnte nicht erstellt werden
if(handle==INVALID_HANDLE)
{
    //--- darüber schreiben und Nummer des Fehlers anzeigen
    PrintFormat("Handle des Indikators iBWMFI für das Paar %s/%s konnte nicht erstellt werden",
                name,
                EnumToString(period),
                GetLastError());
    //--- Arbeit des Indikators ist früher geendet
    return(INIT_FAILED);
};
//--- Das Paar, Symbol/Zeitraumen, auf deren Market Facilitation Index von Bill Williams
short_name=StringFormat("iBWMFI(%s/%s, %s)",name,EnumToString(period),
                        EnumToString(applied_volume));
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- normale Initialisierung des Indikators
return(INIT_SUCCEEDED);
};
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{
    //--- Anzahl der Werte des Indikators iBWMFI zu kopieren
    int values_to_copy;
    //--- Anzahl der berechneten Werte im Indikator finden
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() hat %d zurückgegeben, Fehlercode ist %d",calculated,GetLastError());
        return(0);
    }
};

```

```

//--- wenn dies der erste Start der Berechnung des Indikators ist oder die Anzahl der
//--- oder wenn Sie brauchen den Indikator für zwei oder mehr Balken (was bedeutet, da
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
    {
        //--- wenn Array Tenkan_sen_Buffer größer als die Anzahl der Werte in iBWMFI auf
        //--- sonst kopieren wir weniger als Größe der Indikator-Puffer
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    };
else
{
    //--- dann ist unser Indikator nicht das erste Mal berechnet, und seit dem letzten
    //--- nicht mehr als ein Balken für die Berechnung hinzugefügt war
    values_to_copy=(rates_total-prev_calculated)+1;
};

//--- Die Arrays mit den Werten aus dem Indikator Market Facilitation Index von Bill Williams
//--- Wenn FillArraysFromBuffers false zurückgegeben hat, dann die Daten nicht bereit
    if(!FillArraysFromBuffers(iBWMFIBuffer,iBWMFIColors,handle,values_to_copy)) return false;
//--- Nachricht bilden
    string comm=StringFormat("%s ==> Aktualisierte Werte im Indikator %s: %d",
                                TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                                short_name,
                                values_to_copy);

//--- Hilfsnachrichtung auf dem Chart anzeigen
    Comment(comm);

//--- die Anzahl der Werte im Indikator Market Facilitation Index von Bill Williams setzen
    bars_calculated=calculated;

//--- den Wert prev_calculated für den nächsten Anruf zurückgeben
    return(rates_total);
};

//+-----+
//| Indikator-Puffer aus dem Indikator iBWMFI ausfüllen |
//+-----+

bool FillArraysFromBuffers(double &values[], // Indikator-Puffer der Histogrammswerte
                           double &colors[], // Indikator-Puffer der Histogrammsfarben
                           int ind_handle, // Handle von iBWMFI
                           int amount // Anzahl der Werte, die kopiert werden
                           )
{
    //--- Fehlercode rücksetzen
    ResetLastError();

    //--- Teil des Arrays iBWMFIBuffer mit Werten auf Indikator-Puffer mit Index 0 ausfüllen
    if(CopyBuffer(ind_handle,0,0,amount,values)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iBWMFI kopiert werden, Fehlercode: %d",GetLastError());
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden konnte
        return(false);
    };
};

```



```

//--- Teil des Arrays iBWMFIColors mit Werten auf Indikator-Puffer mit Index 1 ausfüll
    if(CopyBuffer(ind_handle,1,0,amount,colors)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iBWMFI kopiert werden, Fehlercode: %d", GetLastError());
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden konnte
        return(false);
    }
//--- Alles gelang
    return(true);
};

//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
    //--- Das Chart nach der Löschung des Indikators leeren
    Comment("");
};

```

## iMomentum

Gibt das Handle des Indikators Momentum zurück. Hat nur einen Buffer.

```
int iMomentum(
    string          symbol,          // Symbolname
    ENUM_TIMEFRAMES period,          // Periode
    int             mom_period,      // Glättungsperiode
    ENUM_APPLIED_PRICE applied_price // Preistyp oder handle
);
```

### Parameter

*symbol*

[in] symbolname des Instrumentes, dessen Daten für Berechnung des Indikators verwendet werden. NULL bedeutet das laufende Symbol.

*period*

[in] Wert der Periode kann einer der Aufzählungswerte ENUM\_TIMEFRAMES sein, 0 bedeutet das laufende Timeframe.

*mom\_period*

[in] Periode(Anzahl der Bars) für Berechnung der Veränderung des Preises.

*applied\_price*

[in] Der verwendete Preis. Kann eine der Preiskonstanten ENUM\_APPLIED\_PRICE sein oder Handle eines anderen Indikators.

### Rückgabewert

Gibt das Handle des angegebenen technischen Indikators zurück, beim Misserfolg gibt INVALID\_HANDLE zurück. Um Computerspeicher aus nicht genutzten Indikatoren zu befreien, wird die Funktion IndicatorRelease() verwendet, zu deren Indikatorhandle passiert wird.

### Beispiel:

```
//+-----+
//|                                     Demo_iMomentum.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Der Indikator zeigt, wie Daten erhalten werden sollen"
#property description "Indikator-Puffer für den technischen Indikator iMomentum."
#property description "Symbol und Zeitrahmen, in denen der Indikator berechnet wird,"
#property description "sind durch die Parameter symbol und period angegeben."
#property description "Erstellungsweise des Handles ist durch den 'type' Parameter (T)"
#property description "Alle andere Parameter sind wie im normalen Indikator Momentum."

#property indicator_separate_window
```

```

#property indicator_buffers 1
#property indicator_plots 1
//--- plot iMomentum
#property indicator_label1 "iMomentum"
#property indicator_type1 DRAW_LINE
#property indicator_color1 clrDodgerBlue
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//+-----+
//| Aufzählung der Erstellungsweisen des Handles |
//+-----+
enum Creation
{
    Call_iMomentum,      // iMomentum verwenden
    Call_IndicatorCreate  // IndicatorCreate verwenden
};
//--- Eingabeparameter
input Creation          type=Call_iMomentum;      // Funktionstyp
input int               mom_period=14;             // Periode von Momentum
input ENUM_APPLIED_PRICE applied_price=PRICE_CLOSE; // Preistyp
input string            symbol=" ";               // Symbol
input ENUM_TIMEFRAMES   period=PERIOD_CURRENT;    // Zeitrahmen
//--- Indicator-Puffer
double          iMomentumBuffer[];
//--- Eine Variable um Handle des Indikators iMomentum zu speichern
int    handle;
//--- Variable für Speicherung
string name=symbol;
//--- Name des Indikators auf dem Chart
string short_name;
//--- die Anzahl der Werte im Indikator Momentum gespeichert wird
int    bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- Bindung von Array zum Indikator-Puffer
    SetIndexBuffer(0,iMomentumBuffer,INDICATOR_DATA);
    //--- das Symbol, auf das der Indikator gebaut wird, definieren
    name=symbol;
    //--- Leerzeichen aus dem linken und rechten löschen
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- Wenn nach diesem die Länge des String name ist Null
    if(StringLen(name)==0)
    {
        //--- das Symbol aus dem Chart, auf dem der Indikator läuft, nehmen
        name=_Symbol;
    }
}

```

```

    ;}
//--- Erstellen ein Handel des Indikators
if(type==Call_iMomentum)
    handle=iMomentum(name,period,mom_period,applied_price);
else
{
    //--- die Struktur mit Werten von Indikatorparametern ausfüllen
    MqlParam pars[2];
    //--- Periode
    pars[0].type=TYPE_INT;
    pars[0].integer_value=mom_period;
    //--- Preistyp
    pars[1].type=TYPE_INT;
    pars[1].integer_value=applied_price;
    handle=IndicatorCreate(name,period,IND_MOMENTUM,2,pars);
}
//--- Wenn Handle konnte nicht erstellt werden
if(handle==INVALID_HANDLE)
{
    //--- darüber schreiben und Nummer des Fehlers anzeigen
    PrintFormat("Handle des Indikators iMomentum für das Paar %s/%s konnte nicht erst
                name,
                EnumToString(period),
                GetLastError());
    //--- Arbeit des Indikators ist früher geendet
    return(INIT_FAILED);
}
//--- Das Paar von Symbol/Zeitraumen, auf deren Momentum berechnet war, anzeigen
short_name=StringFormat("iMomentum(%s/%s, %d, %s)",name,EnumToString(period),
                        mom_period, EnumToString(applied_price));
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- normale Initialisierung des Indikators
return(INIT_SUCCEEDED);
;}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{
    //--- Anzahl der Werte des Indikators iMomentum zu kopieren

```

```

    int values_to_copy;
    //--- Anzahl der berechneten Werte im Indikator finden
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() hat %d zurückgegeben, Fehlercode ist %d",calculated,GetLastError());
        return(0);
    }
    //--- wenn dies der erste Start der Berechnung des Indikators ist oder die Anzahl der
    //--- oder wenn Sie brauchen den Indikator für zwei oder mehr Balken (was bedeutet, dass
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
    {
        //--- wenn Array iMomentumBuffer größer als die Anzahl der Werte in iMomentum array
        //--- sonst kopieren wir weniger als Größe der Indikator-Puffer
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    }
    else
    {
        //--- dann ist unser Indikator nicht das erste Mal berechnet, und seit dem letzten Mal
        //--- nicht mehr als ein Balken für die Berechnung hinzugefügt war
        values_to_copy=(rates_total-prev_calculated)+1;
    }
    //--- Das Array iMomentumBuffer mit den Werten aus dem Indikator Momentum ausfüllen
    //--- Wenn FillArrayFromBuffer false zurückgegeben hat, dann die Daten nicht bereit sind
    if(!FillArrayFromBuffer(iMomentumBuffer,handle,values_to_copy)) return(0);
    //--- Nachricht bilden
    string comm=StringFormat("%s ==> Aktualisierte Werte im Indikator %s: %d",
                              TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                              short_name,
                              values_to_copy);
    //--- Hilfsnachrichtung auf dem Chart anzeigen
    Comment(comm);
    //--- die Anzahl der Werte im Indikator Momentum speichern
    bars_calculated=calculated;
    //--- den Wert prev_calculated für den nächsten Anruf zurückgeben
    return(rates_total);
}

//+-----+
//| Indikator-Puffer aus dem Indikator iMomentum ausfüllen |
//+-----+

bool FillArrayFromBuffer(double &values[], // Indikator-Puffer der Werte von Momentum
                        int ind_handle,    // Handle des Indikators iMomentum
                        int amount         // Anzahl der Werte, die kopiert werden
                        )
{
    //--- Fehlercode rücksetzen
    ResetLastError();
    //--- Teil des Arrays iMomentumBuffer mit Werten auf Indikator-Puffer mit Index 0 ausfüllen

```

```
if(CopyBuffer(ind_handle,0,0,amount,values)<0)
{
    //-- wenn die Kopie fehlschlägt, Fehlercode anzeigen
    PrintFormat("Daten konnte nicht aus dem Indikator iMomentum kopiert werden, Fehl
    //-- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berech
    return(false);
};
//-- Alles gelang
return(true);
};
//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
    //-- Das Chart nach der Löschung des Indikators leeren
    Comment("");
};
```

## iMFI

Berechnung Money Flow Index.

```
int iMFI(
    string          symbol,          // Symbolname
    ENUM_TIMEFRAMES period,          // Periode
    int             ma_period,        // Glättungsperiode
    ENUM_APPLIED_VOLUME applied_volume // Volumentyp für Berechnung
);
```

### Parameter

*symbol*

[in] Symbolname des Instrumentes, dessen Daten für Berechnung des Indikators verwendet werden. [NULL](#) bedeutet das laufende Symbol.

*period*

[in] Wert der Periode kann einer der Aufzählungswerte [ENUM\\_TIMEFRAMES](#) sein, 0 bedeutet das laufende Timeframe.

*ma\_period*

[in] Periode(Anzahl der Bars) für Berechnung des Indikators.

*applied\_volume*

[in] Das verwendete Volumen. Kann einer der Aufzählungswerte [ENUM\\_APPLIED\\_VOLUME](#) sein.

### Rückgabewert

Gibt das Handle des angegebenen technischen Indikators zurück, beim Misserfolg gibt [INVALID\\_HANDLE](#) zurück. Um Computerspeicher aus nicht genutzten Indikatoren zu befreien, wird die Funktion [IndicatorRelease\(\)](#) verwendet, zu deren Indikatorhandle passiert wird.

### Beispiel:

```
//+-----+
//|                                     Demo_iMFI.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Der Indikator zeigt, wie Daten erhalten werden sollen"
#property description "Indikator-Puffer für den technischen Indikator iMFI."
#property description "Symbol und Zeitrahmen, in denen der Indikator berechnet wird,"
#property description "sind durch die Parameter symbol und period angegeben."
#property description "Erstellungsweise des Handles ist durch den 'type' Parameter (T)"
#property description "Alle andere Parameter sind wie im normalen Money Flow Index."

#property indicator_separate_window
#property indicator_buffers 1
```

```

#property indicator_plots 1
//--- iMFI bauen
#property indicator_label1 "iMFI"
#property indicator_type1 DRAW_LINE
#property indicator_color1 clrDodgerBlue
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//--- Horizontale Ebenen im Indikatorsfenster
#property indicator_level1 20
#property indicator_level2 80
//+-----+
//| Aufzählung der Erstellungsweisen des Handles |
//+-----+
enum Creation
{
    Call_iMFI,          // iMFI verwenden
    Call_IndicatorCreate // IndicatorCreate verwenden
};
//--- Eingabeparameter
input Creation      type=Call_iMFI;          // Funktionstyp
input int           ma_period=14;            // Periode
input ENUM_APPLIED_VOLUME applied_volume=VOLUME_TICK; // Volumentyp
input string        symbol=" ";              // Symbol
input ENUM_TIMEFRAMES period=PERIOD_CURRENT; // Zeitrahmen
//--- Indicator-Puffer
double iMFIBuffer[];
//--- Eine Variable um Handle des Indikators iMFI zu speichern
int handle;
//--- Variable für Speicherung
string name=symbol;
//--- Name des Indikators auf dem Chart
string short_name;
//--- die Anzahl der Werte im Indikator Money Flow Index gespeichert wird
int bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- Bindung von Array zum Indikator-Puffer
    SetIndexBuffer(0,iMFIBuffer,INDICATOR_DATA);
    //--- das Symbol, auf das der Indikator gebaut wird, definieren
    name=symbol;
    //--- Leerzeichen aus dem linken und rechten löschen
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- Wenn nach diesem die Länge des String name ist Null
    if(StringLen(name)==0)
    {

```



```

    //--- das Symbol aus dem Chart, auf dem der Indikator läuft, nehmen
    name=_Symbol;
};

//--- Erstellen ein Handel des Indikators
if(type==Call_iMFI)
    handle=iMFI(name,period,ma_period,applied_volume);
else
{
    //--- die Struktur mit Werten von Indikatorparametern ausfüllen
    MqlParam pars[2];
    //--- Periode
    pars[0].type=TYPE_INT;
    pars[0].integer_value=ma_period;
    //--- Volumentyp
    pars[1].type=TYPE_INT;
    pars[1].integer_value=applied_volume;
    handle=IndicatorCreate(name,period,IND_MFI,2,pars);
};

//--- Wenn Handle konnte nicht erstellt werden
if(handle==INVALID_HANDLE)
{
    //--- darüber schreiben und Nummer des Fehlers anzeigen
    PrintFormat("Handle des Indikators iMFI für das Paar %s/%s konnte nicht erstellt
                name,
                EnumToString(period),
                GetLastError());

    //--- Arbeit des Indikators ist früher geendet
    return(INIT_FAILED);
};

//--- Das Paar, Symbol/Zeitraumen, auf deren Money Flow Index berechnet war, zeigen
short_name=StringFormat("iMFI(%s/%s, %d, %s)",name,EnumToString(period),
                        ma_period, EnumToString(applied_volume));
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- normale Initialisierung des Indikators
return(INIT_SUCCEEDED);
};

//+-----+
//| Custom indicator iteration function |
//+-----+

int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])

```

```

{
//--- Anzahl der Werte des Indikators iMFI zu kopieren
    int values_to_copy;
//--- Anzahl der berechneten Werte im Indikator finden
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() hat %d zurückgegeben, Fehlercode ist %d",calculated,GetLastError());
        return(0);
    }
//--- wenn dies der erste Start der Berechnung des Indikators ist oder die Anzahl der
//--- oder wenn Sie brauchen den Indikator für zwei oder mehr Balken (was bedeutet, dass
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
    {
        //--- wenn Array iMFIBuffer größer als die Anzahl der Werte in iMFI auf symbol/price
        //--- sonst kopieren wir weniger als Größe der Indikator-Puffer
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    }
else
    {
        //--- dann ist unser Indikator nicht das erste Mal berechnet, und seit dem letzten Mal
        //--- nicht mehr als ein Balken für die Berechnung hinzugefügt war
        values_to_copy=(rates_total-prev_calculated)+1;
    }
//--- Das Array iMFIBuffer mit den Werten aus dem Indikator Money Flow Index ausfüllen
//--- Wenn FillArrayFromBuffer false zurückgegeben hat, dann die Daten nicht bereit sind
    if(!FillArrayFromBuffer(iMFIBuffer,handle,values_to_copy)) return(0);
//--- Nachricht bilden
    string comm=StringFormat("%s ==> Aktualisierte Werte im Indikator %s: %d",
                               TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                               short_name,
                               values_to_copy);
//--- Hilfsnachrichtung auf dem Chart anzeigen
    Comment(comm);
//--- die Anzahl der Werte im Indikator Money Flow Index speichern
    bars_calculated=calculated;
//--- den Wert prev_calculated für den nächsten Anruf zurückgeben
    return(rates_total);
};

//+-----+
//| Indikator-Puffer aus dem Indikator iMFI ausfüllen |
//+-----+
bool FillArrayFromBuffer(double &values[], // Indikator-Puffer der Werte von Money Flow Index
                        int ind_handle,    // Handle des Indikators iMFI
                        int amount        // Anzahl der Werte, die kopiert werden
                        )
{
//--- Fehlercode rücksetzen

```

```

    ResetLastError();
    ///--- Teil des Arrays iMFIBuffer mit Werten auf Indikator-Puffer mit Index 0 ausfüllen
    if(CopyBuffer(ind_handle,0,0,amount,values)<0)
    {
        ///--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iMFI kopiert werden, Fehlercode: %d", GetLastError());
        ///--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden konnte
        return(false);
    }
    ///--- Alles gelang
    return(true);
};

//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
    ///--- Das Chart nach der Löschung des Indikators leeren
    Comment("");
};

```

## iMA

Gibt das Handle des Indikators von Mooving Average zurück. Hat nur einen Puffer.

```
int iMA(
    string          symbol,          // Symbolname
    ENUM_TIMEFRAMES period,          // Periode
    int             ma_period,        // Mittelungsperiode
    int             ma_shift,         // horizontale Verschiebung des Indikators
    ENUM_MA_METHOD   ma_method,       // Glättungstyp
    ENUM_APPLIED_PRICE applied_price  // Preistyp oder handle
);
```

### Parameter

*symbol*

[in] Symbolname des Instrumentes, dessen Daten für Berechnung des Indikators verwendet werden. [NULL](#) bedeutet das laufende Symbol.

*period*

[in] Wert der Periode kann einer der Aufzählungswerte [ENUM\\_TIMEFRAMES](#) sein, 0 bedeutet das laufende Timeframe.

*ma\_period*

[in] Mittelungsperiode für Berechnung des Glättungsmittelwertes.

*ma\_shift*

[in] Verschiebung des Indikators in Bezug auf das Preischart.

*ma\_method*

[in] Mittelungsmethode. Kann einer der Werte [ENUM\\_MA\\_METHOD](#) sein.

*applied\_price*

[in] Der verwendete Preis. Kann eine der Preiskonstanten [ENUM\\_APPLIED\\_PRICE](#) oder Handle eines anderen Indikators sein.

### Rückgabewert

Gibt das Handle des angegebenen technischen Indikators zurück, beim Misserfolg gibt [INVALID\\_HANDLE](#) zurück. Um Computerspeicher aus nicht genutzten Indikatoren zu befreien, wird die Funktion [IndicatorRelease\(\)](#) verwendet, zu deren Indikatorhandle passiert wird.

### Beispiel:

```
//+-----+
//|                                     Demo_iMA.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

```

#property description "Der Indikator zeigt, wie Daten erhalten werden sollen"
#property description "Indikator-Puffer für den technischen Indikator iMA."
#property description "Symbol und Zeitrahmen, in denen der Indikator berechnet wird,"
#property description "sind durch die Parameter symbol und period angegeben."
#property description "Erstellungsweise des Handles ist durch den 'type' Parameter (Ty
#property description "Alle andere Parameter sind wie im normalen Indikator Moving Ave

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots 1
//--- iMA bauen
#property indicator_label1 "iMA"
#property indicator_type1 DRAW_LINE
#property indicator_color1 clrRed
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//+-----+
//| Aufzählung der Erstellungsweisen des Handles |
//+-----+
enum Creation
{
    Call_iMA,          // iMA verwenden
    Call_IndicatorCreate // IndicatorCreate verwenden
};
//--- Eingabeparameter
input Creation      type=Call_iMA;          // Funktionstyp
input int           ma_period=10;           // MA Periode
input int           ma_shift=0;             // Verschiebung
input ENUM_MA_METHOD ma_method=MODE_SMA;    // Art der Glättung
input ENUM_APPLIED_PRICE applied_price=PRICE_CLOSE; // Preistyp
input string        symbol=" ";            // Symbol
input ENUM_TIMEFRAMES period=PERIOD_CURRENT; // Zeitrahmen
//--- Indikator-Puffer
double iMABuffer[];
//--- Eine Variable um Handle des Indikators iMA zu speichern
int handle;
//--- Variable für Speicherung
string name=symbol;
//--- Name des Indikators auf dem Chart
string short_name;
//--- die Anzahl der Werte im Indikator Moving Average gespeichert wird
int bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- Bindung von Array zum Indikator-Puffer
    SetIndexBuffer(0,iMABuffer,INDICATOR_DATA);

```

```

//--- Verschiebung definieren
    PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- das Symbol, auf das der Indikator gebaut wird, definieren
    name=symbol;
//--- Leerzeichen aus dem linken und rechten löschen
    StringTrimRight(name);
    StringTrimLeft(name);
//--- Wenn nach diesem die Länge des String name ist Null
    if(StringLen(name)==0)
    {
        //--- das Symbol aus dem Chart, auf dem der Indikator läuft, nehmen
        name=_Symbol;
    }
//--- Erstellen ein Handel des Indikators
    if(type==Call_iMA)
        handle=iMA(name,period,ma_period,ma_shift,ma_method,applied_price);
    else
    {
        //--- die Struktur mit Werten von Indikatorparametern ausfüllen
        MqlParam pars[4];
        //--- Periode
        pars[0].type=TYPE_INT;
        pars[0].integer_value=ma_period;
        //--- Verschiebung
        pars[1].type=TYPE_INT;
        pars[1].integer_value=ma_shift;
        //--- Art der Glättung
        pars[2].type=TYPE_INT;
        pars[2].integer_value=ma_method;
        //--- Preistyp
        pars[3].type=TYPE_INT;
        pars[3].integer_value=applied_price;
        handle=IndicatorCreate(name,period,IND_MA,4,pars);
    }
//--- Wenn Handle konnte nicht erstellt werden
    if(handle==INVALID_HANDLE)
    {
        //--- darüber schreiben und Nummer des Fehlers anzeigen
        PrintFormat("Handle des Indikators iMA für das Paar %s/%s konnte nicht erstellt
                    name,
                    EnumToString(period),
                    GetLastError());
        //--- Arbeit des Indikators ist früher geendet
        return(INIT_FAILED);
    }
//--- Das Paar, Symbol/Zeitraumen, auf deren Moving Average berechnet war, zeigen
    short_name=StringFormat("iMA(%s/%s, %d, %d, %s, %s)",name,EnumToString(period),
                            ma_period, ma_shift,EnumToString(ma_method),EnumToString(ap
    IndicatorSetString(INDICATOR_SHORTNAME,short_name);

```

```

//--- normale Initialisierung des Indikators
    return(INIT_SUCCEEDED);
; }
//+-----+
//| Custom indicator iteration function |
//+-----+

int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
//--- Anzahl der Werte des Indikators iMA zu kopieren
    int values_to_copy;
//--- Anzahl der berechneten Werte im Indikator finden
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() hat %d zurückgegeben, Fehlercode ist %d",calculated,GetLastError());
        return(0);
    }
; }
//--- wenn dies der erste Start der Berechnung des Indikators ist oder die Anzahl der
//--- oder wenn Sie brauchen den Indikator für zwei oder mehr Balken (was bedeutet, dass
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
    {
        //--- wenn Array iMABuffer größer als die Anzahl der Werte in iMA auf symbol/period
        //--- sonst kopieren wir weniger als Größe der Indikator-Puffer
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    }
; }
else
{
    //--- dann ist unser Indikator nicht das erste Mal berechnet, und seit dem letzten Mal
    //--- nicht mehr als ein Balken für die Berechnung hinzugefügt war
    values_to_copy=(rates_total-prev_calculated)+1;
; }

//--- Das Array iMABuffer mit den Werten aus dem Indikator Moving Average ausfüllen
//--- Wenn FillArrayFromBuffer false zurückgegeben hat, dann die Daten nicht bereit sind
    if(!FillArrayFromBuffer(iMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- Nachricht bilden
    string comm=StringFormat("%s ==> Aktualisierte Werte im Indikator %s: %d",
                             TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                             short_name,
                             values_to_copy);

```

```

//--- Hilfsnachrichtung auf dem Chart anzeigen
    Comment(comm);
//--- die Anzahl der Werte im Indikator Moving Average speichern
    bars_calculated=calculated;
//--- den Wert prev_calculated für den nächsten Anruf zurückgeben
    return(rates_total);
; }

//+-----+
//| Indikator-Puffer aus dem Indikator iMA ausfüllen |
//+-----+
bool FillArrayFromBuffer(double &values[], // Indikator-Puffer der Werte von Moving
                        int shift,         // Verschiebung
                        int ind_handle,     // Handle des Indikators iMA
                        int amount         // Anzahl der Werte, die kopiert werden
                        )
{
//--- Fehlercode rücksetzen
    ResetLastError();
//--- Teil des Arrays iMABuffer mit Werten auf Indikator-Puffer mit Index 0 ausfüllen
    if(CopyBuffer(ind_handle,0,-shift,amount,values)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iMA kopiert werden, Fehlercode");
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berech
        return(false);
    }
; }

//--- Alles gelang
    return(true);
; }

//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
//--- Das Chart nach der Löschung des Indikators leeren
    Comment("");
; }

```



## iOsMA

Gibt das Handle des Indikators Moving Average of Oscillator zurück. Oscillator OsMA zeigt Unterschied zwischen den Werten MACD und seiner Signallinie. Hat nur einen Puffer.

```
int iOsMA(
    string          symbol,          // Symbolname
    ENUM_TIMEFRAMES period,          // Periode
    int             fast_ema_period, // Periode von Fast Moving Average
    int             slow_ema_period, // Periode von Slow Moving Average
    int             signal_period,   // Mittelungsperiode der Differenz
    ENUM_APPLIED_PRICE applied_price // Preistyp oder handle
);
```

### Parameter

*symbol*

[in] Symbolname des Instrumentes, dessen Daten für Berechnung des Indikators verwendet werden. [NULL](#) bedeutet das laufende Symbol.

*period*

[in] Wert der Periode kann einer der Aufzählungswerte [ENUM\\_TIMEFRAMES](#) sein, 0 bedeutet das laufende Timeframe.

*fast\_ema\_period*

[in] Mittelungsperiode für Berechnung von Fast Moving Average.

*slow\_ema\_period*

[in] Mittelungsperiode für Berechnung von Slow Moving Average.

*signal\_period*

[in] Mittelungsperiode für Berechnung von Signallinie.

*applied\_price*

[in] Der verwendete Preis. Kann eine der Preiskonstanten [ENUM\\_APPLIED\\_PRICE](#) oder Handle eines anderen Indikators sein.

### Rückgabewert

Gibt das Handle des angegebenen technischen Indikators zurück, beim Misserfolg gibt [INVALID\\_HANDLE](#) zurück. Um Computerspeicher aus nicht genutzten Indikatoren zu befreien, wird die Funktion [IndicatorRelease\(\)](#) verwendet, zu deren Indikatorhandle passiert wird.

### Bemerkung

In einigen Systemen wird dieser Oscillator Histogramm MACD genannt.

### Beispiel:

```
//+-----+
//|                                     Demo_iOsMA.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
```

```

//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Der Indikator zeigt, wie Daten erhalten werden sollen"
#property description "Indikator-Puffer für den technischen Indikator iOsMA."
#property description "Symbol und Zeitrahmen, in denen der Indikator berechnet wird,"
#property description "sind durch die Parameter symbol und period angegeben."
#property description "Erstellungsweise des Handles ist durch den 'type' Parameter (Ty
#property description "Alle andere Parameter sind wie im normalen Indikator Moving Ave

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- iOsMA bauen
#property indicator_label1  "iOsMA"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrSilver
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+-----+
//| Aufzählung der Erstellungsweisen des Handles |
//+-----+
enum Creation
{
    Call_iOsMA,          // iOsMA verwenden
    Call_IndicatorCreate // IndicatorCreate verwenden
};
//--- Eingabeparameter
input Creation      type=Call_iOsMA;          // Funktionstyp
input int           fast_ema_period=12;        // Periode der schnellen MA
input int           slow_ema_period=26;        // Periode der langsamen MA
input int           signal_period=9;           // Mittelungsperiode der Untersc
input ENUM_APPLIED_PRICE applied_price=PRICE_CLOSE; // Preistyp
input string        symbol=" ";               // Symbol
input ENUM_TIMEFRAMES period=PERIOD_CURRENT;  // Zeitrahmen
//--- Indikator-Puffer
double           iOsMABuffer[];
//--- Eine Variable um Handle des Indikators iAMA zu speichern
int              handle;
//--- Variable für Speicherung
string name=symbol;
//--- Name des Indikators auf dem Chart
string short_name;
//--- die Anzahl der Werte im Indikator Moving Average of Oscillator gespeichert wird
int              bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+

```

```

int OnInit()
{
    //--- Bindung von Array zum Indikator-Puffer
    SetIndexBuffer(0,iOsMABuffer,INDICATOR_DATA);
    //--- das Symbol, auf das der Indikator gebaut wird, definieren
    name=symbol;
    //--- Leerzeichen aus dem linken und rechten löschen
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- Wenn nach diesem die Länge des String name ist Null
    if(StringLen(name)==0)
    {
        //--- das Symbol aus dem Chart, auf dem der Indikator läuft, nehmen
        name=_Symbol;
    }
    //--- Erstellen ein Handel des Indikators
    if(type==Call_iOsMA)
        handle=iOsMA(name,period,fast_ema_period,slow_ema_period,signal_period,applied_p
    else
    {
        //--- die Struktur mit Werten von Indikatorparametern ausfüllen
        MqlParam pars[4];
        //--- Schnelle Periode
        pars[0].type=TYPE_INT;
        pars[0].integer_value=fast_ema_period;
        //--- Langsame Periode
        pars[1].type=TYPE_INT;
        pars[1].integer_value=slow_ema_period;
        //--- Mittelungsperiode der Unterschied zwischen der schnellen und langsamen MA
        pars[2].type=TYPE_INT;
        pars[2].integer_value=signal_period;
        //--- Preistyp
        pars[3].type=TYPE_INT;
        pars[3].integer_value=applied_price;
        handle=IndicatorCreate(name,period,IND_OSMA,4,pars);
    }
    //--- Wenn Handle konnte nicht erstellt werden
    if(handle==INVALID_HANDLE)
    {
        //--- darüber schreiben und Nummer des Fehlers anzeigen
        PrintFormat("Handle des Indikators iOsMA für das Paar %s/%s konnte nicht erstell
            name,
            EnumToString(period),
            GetLastError());
        //--- Arbeit des Indikators ist früher geendet
        return(INIT_FAILED);
    }
    //--- Das Paar, Symbol/Zeitraumen, auf deren Moving Average of Oscillator berechnet wa
    short_name=StringFormat("iOsMA(%s/%s,%d,%d,%d,%s)",name,EnumToString(period),

```

```

        fast_ema_period,slow_ema_period,signal_period,EnumToString
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- normale Initialisierung des Indikators
return(INIT_SUCCEEDED);
; }
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
//--- Anzahl der Werte des Indikators iOsMA zu kopieren
int values_to_copy;
//--- Anzahl der berechneten Werte im Indikator finden
int calculated=BarsCalculated(handle);
if(calculated<=0)
{
    PrintFormat("BarsCalculated() hat %d zurückgegeben, Fehlercode ist %d",calculated,GetLastError());
    return(0);
; }
//--- wenn dies der erste Start der Berechnung des Indikators ist oder die Anzahl der
//--- oder wenn Sie brauchen den Indikator für zwei oder mehr Balken (was bedeutet, dass
if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
{
    //--- wenn Array iOsMABuffer größer als die Anzahl der Werte in iOsMA auf symbol
    //--- sonst kopieren wir weniger als Größe der Indikator-Puffer
    if(calculated>rates_total) values_to_copy=rates_total;
    else
        values_to_copy=calculated;
; }
else
{
    //--- dann ist unser Indikator nicht das erste Mal berechnet, und seit dem letzten
    //--- nicht mehr als ein Balken für die Berechnung hinzugefügt war
    values_to_copy=(rates_total-prev_calculated)+1;
; }
//--- Die Arrays mit den Werten aus dem Indikator iOsMA ausfüllen
//--- Wenn FillArrayFromBuffer false zurückgegeben hat, dann die Daten nicht bereit sind
if(!FillArrayFromBuffer(iOsMABuffer,handle,values_to_copy)) return(0);
//--- Nachricht bilden
string comm=StringFormat("%s ==> Aktualisierte Werte im Indikator %s: %d",
                          TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),

```

```

        short_name,
        values_to_copy);
//--- Hilfsnachrichtung auf dem Chart anzeigen
    Comment(comm);
//--- die Anzahl der Werte im Indikator Moving Average of Oscillator speichern
    bars_calculated=calculated;
//--- den Wert prev_calculated für den nächsten Anruf zurückgeben
    return(rates_total);
; }

//+-----+
//| Indikator-Puffer aus dem Indikator iOsMA ausfüllen |
//+-----+
bool FillArrayFromBuffer(double &ama_buffer[], // Indikator-Puffer der Werte von OsMA
                        int ind_handle,        // Handle des Indikators iOsMA
                        int amount             // Anzahl der Werte, die kopiert werden
                        )
{
//--- Fehlercode rücksetzen
    ResetLastError();
//--- Teil des Arrays iOsMABuffer mit Werten auf Indikator-Puffer mit Index 0 ausfüllen
    if(CopyBuffer(ind_handle,0,0,amount,ama_buffer)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iOsMA kopiert werden, Fehlercode: %d", GetLastError());
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden konnte
        return(false);
    }
//--- Alles gelang
    return(true);
; }

//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
//--- Das Chart nach der Löschung des Indikators leeren
    Comment("");
; }

```

## iMACD

Gibt das Handle des Indikators Moving Averages Convergence/Divergence zurück. In den Systemen, wo OsMA als Histogramm MACD bezeichnet wird, wird dieser Indikator als zwei Linien dargestellt. Im Kundenterminal wird Konvergenz/Divergenz von Moving Averages als Histogramm dargestellt.

```
int iMACD(
    string          symbol,          // Symbolname
    ENUM_TIMEFRAMES period,          // Periode
    int             fast_ema_period,  // Periode für Berechnung von Fast average
    int             slow_ema_period,  // Periode für Berechnung von Slow average
    int             signal_period,    // Periode für Mittelung ihrer Differenz
    ENUM_APPLIED_PRICE applied_price  // Preistyp oder handle
);
```

### Parameter

*symbol*

[in] Symbolname des Instrumentes dessen Daten für Berechnung des Indikators verwendet werden. [NULL](#) bedeutet das laufende Symbol.

*period*

[in] Wert der Periode kann einer der Aufzählungswerte [ENUM\\_TIMEFRAMES](#) sein, 0 bedeutet das laufende Timeframe.

*fast\_ema\_period*

[in] Mittelungsperiode für Berechnung von Fast Moving Average.

*slow\_ema\_period*

[in] Mittelungsperiode für Berechnung von Slow Moving Average.

*signal\_period*

[in] Mittelungsperiode für Berechnung von Signallinie.

*applied\_price*

[in] Der verwendete Preis. Kann einer der Preiskonstanten [ENUM\\_APPLIED\\_PRICE](#) oder Handle eines anderen Indikators sein.

### Rückgabewert

Gibt das Handle des angegebenen technischen Indikators, beim Misserfolg gibt [INVALID\\_HANDLE](#) zurück. Um Computerspeicher aus nicht genutzten Indikatoren zu befreien, wird die Funktion [IndicatorRelease\(\)](#) verwendet, zu deren Indikatorhandle passiert wird.

### Bemerkung

Puffernummern: 0 - MAIN\_LINE, 1 - SIGNAL\_LINE.

### Beispiel:

```
//+-----+
//|                                     Demo_iMACD.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
```

```

//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Der Indikator zeigt, wie Daten erhalten werden sollen"
#property description "Indikator-Puffer für den technischen Indikator iMACD."
#property description "Symbol und Zeitrahmen, in denen der Indikator berechnet wird,"
#property description "sind durch die Parameter symbol und period angegeben."
#property description "Erstellungsweise des Handles ist durch den 'type' Parameter (Ty
#property description "Alle andere Parameter sind wie im normalen Indikator MACD."

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   2
//--- MACD bauen
#property indicator_label1  "MACD"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrSilver
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- Signal bauen
#property indicator_label2  "Signal"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_DOT
#property indicator_width2  1
//+-----+
//| Aufzählung der Erstellungsweisen des Handles |
//+-----+
enum Creation
{
    Call_iMACD,          // iMACD verwenden
    Call_IndicatorCreate  // IndicatorCreate verwenden
};
//--- Eingabeparameter
input Creation          type=Call_iMACD;          // Funktionstyp
input int               fast_ema_period=12;        // Periode der schnellen MA
input int               slow_ema_period=26;        // Periode der langsamen MA
input int               signal_period=9;          // Mittelungsperiode der Untersc
input ENUM_APPLIED_PRICE applied_price=PRICE_CLOSE; // Preistyp
input string            symbol=" ";               // Symbol
input ENUM_TIMEFRAMES   period=PERIOD_CURRENT;    // Zeitrahmen
//--- Indikator-Puffer
double      MACDBuffer[];
double      SignalBuffer[];
//--- Eine Variable um Handle des Indikators iMACD zu speichern
int         handle;
//--- Variable für Speicherung

```

```

string name=symbol;
//--- Name des Indikators auf dem Chart
string short_name;
//--- die Anzahl der Werte im Indikator Moving Averages Convergence/Divergence gespeichert
int bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- Bindung von Arrays zu den Indikator-Puffern
SetIndexBuffer(0,MACDBuffer,INDICATOR_DATA);
SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
//--- das Symbol, auf das der Indikator gebaut wird, definieren
name=symbol;
//--- Leerzeichen aus dem linken und rechten löschen
StringTrimRight(name);
StringTrimLeft(name);
//--- Wenn nach diesem die Länge des String name ist Null
if(StringLen(name)==0)
{
//--- das Symbol aus dem Chart, auf dem der Indikator läuft, nehmen
name=_Symbol;
};
//--- Erstellen ein Handel des Indikators
if(type==Call_iMACD)
handle=iMACD(name,period,fast_ema_period,slow_ema_period,signal_period,applied_price);
else
{
//--- die Struktur mit Werten von Indikatorparametern ausfüllen
MqlParam pars[4];
//--- Schnelle Periode
pars[0].type=TYPE_INT;
pars[0].integer_value=fast_ema_period;
//--- Langsame Periode
pars[1].type=TYPE_INT;
pars[1].integer_value=slow_ema_period;
//--- Mittelungsperiode der Unterschied zwischen der schnellen und langsamen MA
pars[2].type=TYPE_INT;
pars[2].integer_value=signal_period;
//--- Preistyp
pars[3].type=TYPE_INT;
pars[3].integer_value=applied_price;
handle=IndicatorCreate(name,period,IND_MACD,4,pars);
};
//--- Wenn Handle konnte nicht erstellt werden
if(handle==INVALID_HANDLE)
{
//--- darüber schreiben und Nummer des Fehlers anzeigen

```



```

        PrintFormat("Handle des Indikators iMACD für das Paar %s/%s konnte nicht erstellt
                    name,
                    EnumToString(period),
                    GetLastError());

        //-- Arbeit des Indikators ist früher geendet
        return(INIT_FAILED);

    ;}

    //-- Das Paar, Symbol/Zeitraumen, auf deren Moving Averages Convergence/Divergence be
    short_name=StringFormat("iMACD(%s/%s,%d,%d,%d,%s)",name,EnumToString(period),
                            fast_ema_period,slow_ema_period,signal_period,EnumToString
    IndicatorSetString(INDICATOR_SHORTNAME,short_name);
    //-- normale Initialisierung des Indikators
    return(INIT_SUCCEEDED);

    ;}

    //+-----+
    //| Custom indicator iteration function |
    //+-----+

    int OnCalculate(const int rates_total,
                    const int prev_calculated,
                    const datetime &time[],
                    const double &open[],
                    const double &high[],
                    const double &low[],
                    const double &close[],
                    const long &tick_volume[],
                    const long &volume[],
                    const int &spread[])

    {
        //-- Anzahl der Werte des Indikators iMACD zu kopieren
        int values_to_copy;
        //-- Anzahl der berechneten Werte im Indikator finden
        int calculated=BarsCalculated(handle);
        if(calculated<=0)
        {
            PrintFormat("BarsCalculated() hat %d zurückgegeben, Fehlercode ist %d",calculated,GetLastError());
            return(0);
        }
        ;}

        //-- wenn dies der erste Start der Berechnung des Indikators ist oder die Anzahl der
        //-- oder wenn Sie brauchen den Indikator für zwei oder mehr Balken (was bedeutet, da
        if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
        {
            //-- wenn Array MACDBuffer größer als die Anzahl der Werte in iMACD auf symbol,
            //-- sonst kopieren wir weniger als Größe der Indikator-Puffer
            if(calculated>rates_total) values_to_copy=rates_total;
            else
                values_to_copy=calculated;
        }
        ;}
    else
    {
        //-- dann ist unser Indikator nicht das erste Mal berechnet, und seit dem letzte

```

```

        //--- nicht mehr als ein Balken für die Berechnung hinzugefügt war
        values_to_copy=(rates_total-prev_calculated)+1;
    };
//--- Die Arrays mit den Werten aus dem Indikator iMACD ausfüllen
//--- Wenn FillArraysFromBuffers false zurückgegeben hat, dann die Daten nicht bereit
    if(!FillArraysFromBuffers(MACDBuffer,SignalBuffer,handle,values_to_copy)) return(0);
//--- Nachricht bilden
    string comm=StringFormat("%s ==> Aktualisierte Werte im Indikator %s: %d",
                               TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                               short_name,
                               values_to_copy);
//--- Hilfsnachrichtung auf dem Chart anzeigen
    Comment(comm);
//--- die Anzahl der Werte im Indikator Moving Averages Convergence/Divergence speichern
    bars_calculated=calculated;
//--- den Wert prev_calculated für den nächsten Anruf zurückgeben
    return(rates_total);
};
//+-----+
//| Indikator-Puffer aus dem Indikator iMACD ausfüllen |
//+-----+
bool FillArraysFromBuffers(double &macd_buffer[], // Indikator-Puffer der Werte vor
                           double &signal_buffer[], // Indikator-Puffer von MACD Signal
                           int ind_handle, // Handle des Indikators iMACD
                           int amount // Anzahl der Werte, die kopiert werden
                           )
{
    //--- Fehlercode rücksetzen
    ResetLastError();
//--- Teil des Arrays iMACDBuffer mit Werten auf Indikator-Puffer mit Index 0 ausfüllen
    if(CopyBuffer(ind_handle,0,0,amount,macd_buffer)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iMACD kopiert werden, Fehlercode: %d",GetLastError());
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden konnte
        return(false);
    };
//--- Teil des Arrays SignalBuffer mit Werten auf Indikator-Puffer mit Index 1 ausfüllen
    if(CopyBuffer(ind_handle,1,0,amount,signal_buffer)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iMACD kopiert werden, Fehlercode: %d",GetLastError());
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden konnte
        return(false);
    };
//--- Alles gelang
    return(true);
};

```

```
//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
//--- Das Chart nach der Löschung des Indikators leeren
    Comment("");
};
```

## iOBV

Gibt das Handle des Indikators On Balance Volume zurück. Hat nur einen Puffer.

```
int iOBV(
    string          symbol,          // Symbolname
    ENUM_TIMEFRAMES period,          // Periode
    ENUM_APPLIED_VOLUME applied_volume // Volumentyp für Berechnung
);
```

### Parameter

*symbol*

[in] Symbolname des Instrumentes, dessen Daten für Berechnung des Indikators verwendet werden. [NULL](#) bedeutet das laufende Symbol.

*period*

[in] Wert der Periode kann einer der Aufzählungswerte [ENUM\\_TIMEFRAMES](#) sein, 0 bedeutet das laufende Timeframe.

*applied\_volume*

[in] The volume used. Can be any of the [ENUM\\_APPLIED\\_VOLUME](#) values.

### Rückgabewert

Gibt handle des angegebenen technischen Indikators zurück, beim Misserfolg gibt [INVALID\\_HANDLE](#) zurück. Um Computerspeicher aus nicht genutzten Indikatoren zu befreien, wird die Funktion [IndicatorRelease\(\)](#) verwendet, zu deren Indikatorhandle passiert wird.

### Beispiel:

```
//+-----+
//|                                     Demo_iOBV.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Der Indikator zeigt, wie Daten erhalten werden sollen"
#property description "Indikator-Puffer für den technischen Indikator iOBV."
#property description "Symbol und Zeitrahmen, in denen der Indikator berechnet wird,"
#property description "sind durch die Parameter symbol und period angegeben."
#property description "Erstellungsweise des Handles ist durch den 'type' Parameter (Ty

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots 1
//--- iOBV
#property indicator_label1 "iOBV"
#property indicator_type1  DRAW_LINE
#property indicator_color1 clrLightSeaGreen
```

```

#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//+-----+
//| Aufzählung der Erstellungsweisen des Handles |
//+-----+
enum Creation
{
    Call_iOBV ,           // iOBV verwenden
    Call_IndicatorCreate  // IndicatorCreate verwenden
};
//--- Eingabeparameter
input Creation          type=Call_iOBV;           // Funktionstyp
input ENUM_APPLIED_VOLUME applied_volume=VOLUME_TICK; // Volumentyp
input string            symbol=" ";               // Symbol
input ENUM_TIMEFRAMES   period=PERIOD_CURRENT;    // Zeitrahmen
//--- Indicator-Puffer
double iOBVBuffer[];
//--- Eine Variable um Handle des Indikators iOBV zu speichern
int handle;
//--- Variable für Speicherung
string name=symbol;
//--- Name des Indikators auf dem Chart
string short_name;
//--- die Anzahl der Werte im Indikator On Balance Volume gespeichert wird
int bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- Bindung von Array zum Indikator-Puffer
    SetIndexBuffer(0,iOBVBuffer,INDICATOR_DATA);
    //--- das Symbol, auf das der Indikator gebaut wird, definieren
    name=symbol;
    //--- Leerzeichen aus dem linken und rechten löschen
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- Wenn nach diesem die Länge des String name ist Null
    if(StringLen(name)==0)
    {
        //--- das Symbol aus dem Chart, auf dem der Indikator läuft, nehmen
        name=_Symbol;
    }
    //--- Erstellen ein Handel des Indikators
    if(type==Call_iOBV)
        handle=iOBV(name,period,applied_volume);
    else
    {
        //--- die Struktur mit Werten von Indikatorparametern ausfüllen

```

```

    MqlParam pars[1];
    //--- Volumentyp
    pars[0].type=TYPE_INT;
    pars[0].integer_value=applied_volume;
    handle=IndicatorCreate(name,period,IND_OBV,1,pars);
; }
//--- Wenn Handle konnte nicht erstellt werden
if(handle==INVALID_HANDLE)
{
    //--- darüber schreiben und Nummer des Fehlers anzeigen
    PrintFormat("Handle des Indikators iOBV für das Paar %s/%s konnte nicht erstellt
                name,
                EnumToString(period),
                GetLastError());
    //--- Arbeit des Indikators ist früher geendet
    return(INIT_FAILED);
; }
//--- Das Paar, Symbol/Zeitraumen, auf deren On Balance Volume berechnet war, zeigen
short_name=StringFormat("iOBV(%s/%s, %s)",name,EnumToString(period),
                        EnumToString(applied_volume));
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- normale Initialisierung des Indikators
return(INIT_SUCCEEDED);
; }
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{
    //--- Anzahl der Werte des Indikators iOBV zu kopieren
    int values_to_copy;
    //--- Anzahl der berechneten Werte im Indikator finden
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() hat %d zurückgegeben, Fehlercode ist %d",calculated,GetLastError());
        return(0);
    ; }
    //--- wenn dies der erste Start der Berechnung des Indikators ist oder die Anzahl der
    //--- oder wenn Sie brauchen den Indikator für zwei oder mehr Balken (was bedeutet, da

```

```

    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
    {
        //--- wenn Array iOBVBuffer größer als die Anzahl der Werte in iOBV auf symbol/p
        //--- sonst kopieren wir weniger als Größe der Indikator-Puffer
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    }
    else
    {
        //--- dann ist unser Indikator nicht das erste Mal berechnet, und seit dem letzten Mal
        //--- nicht mehr als ein Balken für die Berechnung hinzugefügt war
        values_to_copy=(rates_total-prev_calculated)+1;
    }

    //--- Die Arrays mit den Werten aus dem Indikator iOBV ausfüllen
    //--- Wenn FillArrayFromBuffer false zurückgegeben hat, dann die Daten nicht bereit sein
    if(!FillArrayFromBuffer(iOBVBuffer,handle,values_to_copy)) return(0);
    //--- Nachricht bilden
    string comm=StringFormat("%s ==> Aktualisierte Werte im Indikator %s: %d",
                               TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                               short_name,
                               values_to_copy);
    //--- Hilfsnachrichtung auf dem Chart anzeigen
    Comment(comm);
    //--- die Anzahl der Werte im Indikator On Balance Volume speichern
    bars_calculated=calculated;
    //--- den Wert prev_calculated für den nächsten Anruf zurückgeben
    return(rates_total);
};

//+-----+
//| Indikator-Puffer aus dem Indikator iOBV ausfüllen |
//+-----+
bool FillArrayFromBuffer(double &obv_buffer[], // Indikator-Puffer der Werte von OBV
                        int ind_handle,        // Handle des Indikators iOBV
                        int amount            // Anzahl der Werte, die kopiert werden
                        )
{
    //--- Fehlercode rücksetzen
    ResetLastError();
    //--- Teil des Arrays iOBVBuffer mit Werten auf Indikator-Puffer mit Index 0 ausfüllen
    if(CopyBuffer(ind_handle,0,0,amount,obv_buffer)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iOBV kopiert werden, Fehlercode: %d",GetLastError());
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet wurde
        return(false);
    }
    //--- Alles gelang
    return(true);
};

```

```
//+-----+  
//| Indicator deinitialization function |  
//+-----+  
void OnDeinit(const int reason)  
{  
//--- Das Chart nach der Löschung des Indikators leeren  
    Comment("");  
};
```



## iSAR

Gibt das Handle des Indikators Parabolic Stop and Reverse system zurück. Hat nur einen Puffer.

```
int iSAR(
    string          symbol,      // Symbolname
    ENUM_TIMEFRAMES period,     // Periode
    double          step,       // Schritt von Geschwindigkeitsinkrement - Beschleunigung
    double          maximum     // Maximales Stoplevel
);
```

### Parameter

*symbol*

[in] Symbolname des Instrumentes, dessen Daten für Berechnung des Indikators verwendet werden. [NULL](#) bedeutet das laufende Symbol.

*period*

[in] Wert der Periode kann einer der Aufzählungswerte [ENUM\\_TIMEFRAMES](#) sein, 0 bedeutet das laufende Timeframe.

*step*

[in] Inkrement von Stoplevel, gewöhnlich 0.02.

*maximum*

[in] Maximales Stoplevel, gewöhnlich 0.2.

### Rückgabewert

Gibt das Handle des angegebenen technischen Indikators zurück, beim Misserfolg gibt [INVALID\\_HANDLE](#) zurück. Um Computerspeicher aus nicht genutzten Indikatoren zu befreien, wird die Funktion [IndicatorRelease\(\)](#) verwendet, zu deren Indikatorhandle passiert wird.

### Beispiel:

```
//+-----+
//|                                     Demo_iSAR.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Der Indikator zeigt, wie Daten erhalten werden sollen"
#property description "Indikator-Puffer für den technischen Indikator iSAR."
#property description "Symbol und Zeitrahmen, in denen der Indikator berechnet wird,"
#property description "sind durch die Parameter symbol und period angegeben."
#property description "Erstellungsweise des Handles ist durch den 'type' Parameter (Typ)"
#property description "Alle andere Parameter sind wie im normalen Indikator Parabolic"

#property indicator_chart_window
#property indicator_buffers 1
```

```

#property indicator_plots 1
//--- iSAR bauen
#property indicator_label1 "iSAR"
#property indicator_type1 DRAW_ARROW
#property indicator_color1 clrBlue
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//+-----+
//| Aufzählung der Erstellungsweisen des Handles |
//+-----+
enum Creation
{
    Call_iSAR,          // iSAR verwenden
    Call_IndicatorCreate // IndicatorCreate verwenden
};
//--- Eingabeparameter
input Creation      type=Call_iSAR;          // Funktionstyp
input double        step=0.02;               // Schritt - die Beschleunigung
input double        maximum=0.2;             // Maximalwert des Schritts
input string        symbol=" ";              // Symbol
input ENUM_TIMEFRAMES period=PERIOD_CURRENT; // Zeitrahmen
//--- Indikator-Puffer
double iSARBuffer[];
//--- Eine Variable um Handle des Indikators iSAR zu speichern
int handle;
//--- Variable für Speicherung
string name=symbol;
//--- Name des Indikators auf dem Chart
string short_name;
//--- die Anzahl der Werte im Indikator Parabolic SAR gespeichert wird
int bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- Bindung von Array zum Indikator-Puffer
    SetIndexBuffer(0,iSARBuffer,INDICATOR_DATA);
    //--- Code des Symbols aus Reihe von Wingdings für PLOT_ARROW Eigenschaft definieren
    PlotIndexSetInteger(0,PLOT_ARROW,159);
    //--- das Symbol, auf das der Indikator gebaut wird, definieren
    name=symbol;
    //--- Leerzeichen aus dem linken und rechten löschen
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- Wenn nach diesem die Länge des String name ist Null
    if(StringLen(name)==0)
    {
        //--- das Symbol aus dem Chart, auf dem der Indikator läuft, nehmen

```

```

        name=_Symbol;
    };
//--- Erstellen ein Handel des Indikators
    if(type==Call_iSAR)
        handle=iSAR(name,period,step,maximum);
    else
    {
        //--- die Struktur mit Werten von Indikatorparametern ausfüllen
        MqlParam pars[2];
        //--- Schrittwert
        pars[0].type=TYPE_DOUBLE;
        pars[0].double_value=step;
        //--- ser Grenzwert des Schritts, der in den Berechnungen verwendet werden kann
        pars[1].type=TYPE_DOUBLE;
        pars[1].double_value=maximum;
        handle=IndicatorCreate(name,period,IND_SAR,2,pars);
    };
//--- Wenn Handle konnte nicht erstellt werden
    if(handle==INVALID_HANDLE)
    {
        //--- darüber schreiben und Nummer des Fehlers anzeigen
        PrintFormat("Handle des Indikators iSAR für das Paar %s/%s konnte nicht erstellt
                    name,
                    EnumToString(period),
                    GetLastError());
        //--- Arbeit des Indikators ist früher geendet
        return(INIT_FAILED);
    };
//--- Das Paar, Symbol/Zeitraumen, auf deren Parabolic SAR berechnet war, zeigen
    short_name=StringFormat("iSAR(%s/%s, %G, %G)",name,EnumToString(period),
                            step,maximum);
    IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- normale Initialisierung des Indikators
    return(INIT_SUCCEEDED);
};
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
               const int prev_calculated,
               const datetime &time[],
               const double &open[],
               const double &high[],
               const double &low[],
               const double &close[],
               const long &tick_volume[],
               const long &volume[],
               const int &spread[])
{

```

```

//--- Anzahl der Werte des Indikators iSAR zu kopieren
int values_to_copy;
//--- Anzahl der berechneten Werte im Indikator finden
int calculated=BarsCalculated(handle);
if(calculated<=0)
{
    PrintFormat("BarsCalculated() hat %d zurückgegeben, Fehlercode ist %d",calculate
    return(0);
};
//--- wenn dies der erste Start der Berechnung des Indikators ist oder die Anzahl der
//--- oder wenn Sie brauchen den Indikator für zwei oder mehr Balken (was bedeutet, da
if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
{
    //--- wenn Array iSARBuffer größer als die Anzahl der Werte in iSAR auf symbol
    //--- sonst kopieren wir weniger als Größe der Indikator-Puffer
    if(calculated>rates_total) values_to_copy=rates_total;
    else
        values_to_copy=calculated;
};
else
{
    //--- dann ist unser Indikator nicht das erste Mal berechnet, und seit dem letzten
    //--- nicht mehr als ein Balken für die Berechnung hinzugefügt war
    values_to_copy=(rates_total-prev_calculated)+1;
};
//--- Die Arrays mit den Werten aus dem Indikator iSAR ausfüllen
//--- Wenn FillArrayFromBuffer false zurückgegeben hat, dann die Daten nicht bereit sind
if(!FillArrayFromBuffer(iSARBuffer,handle,values_to_copy)) return(0);
//--- Nachricht bilden
string comm=StringFormat("%s ==> Aktualisierte Werte im Indikator %s: %d",
                           TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                           short_name,
                           values_to_copy);
//--- Hilfsnachrichtung auf dem Chart anzeigen
Comment(comm);
//--- die Anzahl der Werte im Indikator Parabolic SAR speichern
bars_calculated=calculated;
//--- den Wert prev_calculated für den nächsten Anruf zurückgeben
return(rates_total);
};
//+-----+
//| Indikator-Puffer aus dem Indikator iSAR ausfüllen |
//+-----+
bool FillArrayFromBuffer(double &sar_buffer[], // Indikator-Puffer der Werte von Parabolic SAR
                        int ind_handle,        // Handle des Indikators iSAR
                        int amount            // Anzahl der Werte, die kopiert werden
                        )
{
    //--- Fehlercode rücksetzen
    ResetLastError();

```

```

//--- Teil des Arrays iSARBuffer mit Werten auf Indikator-Puffer mit Index 0 ausfüllen
if(CopyBuffer(ind_handle,0,0,amount,sar_buffer)<0)
{
    //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
    PrintFormat("Daten konnte nicht aus dem Indikator iSAR kopiert werden, Fehlercode: %d", GetLastError());
    //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden konnte
    return(false);
};
//--- Alles gelang
return(true);
};
//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
    //--- Das Chart nach der Löschung des Indikators leeren
    Comment("");
};

```

## iRSI

Gibt das Handle des Indikators Relative Strength Index zurück. Hat nur einen Puffer.

```
int iRSI(
    string          symbol,          // Symbolname
    ENUM_TIMEFRAMES period,          // Periode
    int             ma_period,        // Mittelungsperiode
    ENUM_APPLIED_PRICE applied_price // Preistyp oder Handle
);
```

### Parameter

*symbol*

[in] Symbolname des Instrumentes, dessen Daten für Berechnung des Indikators verwendet werden. NULL bedeutet das laufende Symbol.

*period*

[in] Wert der Periode kann einer der Aufzählungswerte ENUM\_TIMEFRAMES sein, 0 bedeutet das laufende Timeframe.

*ma\_period*

[in] Mittelungsperiode für Berechnung des Indexes.

*applied\_price*

[in] Der verwendete Preis. Kann eine der Preiskonstanten ENUM\_APPLIED\_PRICE oder Handle eines anderen Indikators sein.

### Rückgabewert

Gibt das Handle des angegebenen technischen Indikators zurück, beim Misserfolg gibt INVALID\_HANDLE zurück. Um Computerspeicher aus nicht genutzten Indikatoren zu befreien, wird die Funktion IndicatorRelease() verwendet, zu deren Indikatorhandle passiert wird.

### Beispiel:

```
//+-----+
//|                                     Demo_iRSI.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Der Indikator zeigt, wie Daten erhalten werden sollen"
#property description "Indikator-Puffer für den technischen Indikator iRSI."
#property description "Symbol und Zeitrahmen, in denen der Indikator berechnet wird,"
#property description "sind durch die Parameter symbol und period angegeben."
#property description "Erstellungsweise des Handles ist durch den 'type' Parameter (T)"
#property description "Alle andere Parameter sind wie im normalen Relative Strength I"

#property indicator_separate_window
```

```

#property indicator_buffers 1
#property indicator_plots 1
//--- iRSI bauen
#property indicator_label1 "iRSI"
#property indicator_type1 DRAW_LINE
#property indicator_color1 clrDodgerBlue
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//--- Limits, um die Werte im Indikatorsfenster anzuzeigen
#property indicator_maximum 100
#property indicator_minimum 0
//--- Horizontale Ebenen im Indikatorsfenster
#property indicator_level1 70.0
#property indicator_level2 30.0
//+-----+
//| Aufzählung der Erstellungsweisen des Handles |
//+-----+
enum Creation
{
    Call_iRSI,          // iRSI verwenden
    Call_IndicatorCreate // IndicatorCreate verwenden
};
//--- Eingabeparameter
input Creation      type=Call_iRSI;          // Funktionstyp
input int           ma_period=14;            // MA Periode
input ENUM_APPLIED_PRICE applied_price=PRICE_CLOSE; // Preistyp
input string        symbol=" ";              // Symbol
input ENUM_TIMEFRAMES period=PERIOD_CURRENT; // Zeitrahmen
//--- Indicator-Puffer
double iRSIBuffer[];
//--- Eine Variable um Handle des Indikators iRSI zu speichern
int handle;
//--- Variable für Speicherung
string name=symbol;
//--- Name des Indikators auf dem Chart
string short_name;
//--- die Anzahl der Werte im Indikator Relative Strength Index gespeichert wird
int bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- Bindung von Array zum Indikator-Puffer
    SetIndexBuffer(0,iRSIBuffer,INDICATOR_DATA);
    //--- das Symbol, auf das der Indikator gebaut wird, definieren
    name=symbol;
    //--- Leerzeichen aus dem linken und rechten löschen
    StringTrimRight(name);

```

```

StringTrimLeft(name);
//--- Wenn nach diesem die Länge des String name ist Null
if(StringLen(name)==0)
{
    //--- das Symbol aus dem Chart, auf dem der Indikator läuft, nehmen
    name=_Symbol;
};
//--- Erstellen ein Handel des Indikators
if(type==Call_iRSI)
    handle=iRSI(name,period,ma_period,applied_price);
else
{
    //--- die Struktur mit Werten von Indikatorparametern ausfüllen
    MqlParam pars[2];
    //--- MA Periode
    pars[0].type=TYPE_INT;
    pars[0].integer_value=ma_period;
    //--- ser Grenzwert des Schritts, der in den Berechnungen verwendet werden kann
    pars[1].type=TYPE_INT;
    pars[1].integer_value=applied_price;
    handle=IndicatorCreate(name,period,IND_RSI,2,pars);
};
//--- Wenn Handle konnte nicht erstellt werden
if(handle==INVALID_HANDLE)
{
    //--- darüber schreiben und Nummer des Fehlers anzeigen
    PrintFormat("Handle des Indikators iRSI für das Paar %s/%s konnte nicht erstellt
                name,
                EnumToString(period),
                GetLastError());
    //--- Arbeit des Indikators ist früher geendet
    return(INIT_FAILED);
};
//--- Das Paar, Symbol/Zeitraumen, auf deren Relative Strength Index berechnet war, ze
short_name=StringFormat("iRSI(%s/%s, %d, %d)",name,EnumToString(period),
                        ma_period,applied_price);
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- normale Initialisierung des Indikators
return(INIT_SUCCEEDED);
};
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],

```



```

        const double &close[],
        const long &tick_volume[],
        const long &volume[],
        const int &spread[])
    {
//--- Anzahl der Werte des Indikators iRSI zu kopieren
        int values_to_copy;
//--- Anzahl der berechneten Werte im Indikator finden
        int calculated=BarsCalculated(handle);
        if(calculated<=0)
        {
            PrintFormat("BarsCalculated() hat %d zurückgegeben, Fehlercode ist %d",calculated,GetLastError());
            return(0);
        };
//--- wenn dies der erste Start der Berechnung des Indikators ist oder die Anzahl der
//--- oder wenn Sie brauchen den Indikator für zwei oder mehr Balken (was bedeutet, dass
        if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
        {
            //--- wenn Array iRSIBuffer größer als die Anzahl der Werte in iRSI auf symbol/price
            //--- sonst kopieren wir weniger als Größe der Indikator-Puffer
            if(calculated>rates_total) values_to_copy=rates_total;
            else
                values_to_copy=calculated;
        };
        else
        {
            //--- dann ist unser Indikator nicht das erste Mal berechnet, und seit dem letzten Mal
            //--- nicht mehr als ein Balken für die Berechnung hinzugefügt war
            values_to_copy=(rates_total-prev_calculated)+1;
        };
//--- Das Array mit den Werten aus dem Indikator iRSI ausfüllen
//--- Wenn FillArrayFromBuffer false zurückgegeben hat, dann die Daten nicht bereit sind
        if(!FillArrayFromBuffer(iRSIBuffer,handle,values_to_copy)) return(0);
//--- Nachricht bilden
        string comm=StringFormat("%s ==> Aktualisierte Werte im Indikator %s: %d",
                                   TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                                   short_name,
                                   values_to_copy);
//--- Hilfsnachrichtung auf dem Chart anzeigen
        Comment(comm);
//--- die Anzahl der Werte im Indikator Relative Strength Index speichern
        bars_calculated=calculated;
//--- den Wert prev_calculated für den nächsten Anruf zurückgeben
        return(rates_total);
    };
//+-----+
//| Indikator-Puffer aus dem Indikator iRSI ausfüllen |
//+-----+
bool FillArrayFromBuffer(double &rsi_buffer[], // Indikator-Puffer der Werte von Relative Strength Index
                        int ind_handle,        // Handle des Indikators iSAR

```

```

        int amount          // Anzahl der Werte, die kopiert werden
    )

    {
    //--- Fehlercode zurücksetzen
        ResetLastError();
    //--- Teil des Arrays iRSIBuffer mit Werten auf Indikator-Puffer mit Index 0 ausfüllen
        if(CopyBuffer(ind_handle,0,0,amount,rsi_buffer)<0)
        {
            //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
            PrintFormat("Daten konnte nicht aus dem Indikator iRSI kopiert werden, Fehlercode: %d", GetLastError());
            //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden konnte
            return(false);
        }
    //--- Alles gelang
        return(true);
    }

    //+-----+
    //| Indicator deinitialization function |
    //+-----+
    void OnDeinit(const int reason)
    {
    //--- Das Chart nach der Löschung des Indikators leeren
        Comment("");
    }

```

## iRVI

Gibt das Handle des Indikators Relative Vigor Index zurück.

```
int iRVI(
    string      symbol,      // Symbolname
    ENUM_TIMEFRAMES period,  // Periode
    int         ma_period    // Mittelungsperiode
);
```

### Parameter

*symbol*

[in] Symbolname des Instrumentes, dessen Daten für Berechnung des Indikators verwendet werden. [NULL](#) bedeutet das laufende Symbol.

*period*

[in] Wert der Periode kann einer der Aufzählungswerte [ENUM\\_TIMEFRAMES](#) sein, 0 bedeutet das laufende Timeframe.

*ma\_period*

[in] Mittelungsperiode für Berechnung des Indexes.

### Rückgabewert

Gibt das Handle des angegebenen technischen Indikators zurück, beim Misserfolg gibt [INVALID\\_HANDLE](#) zurück. Um Computerspeicher aus nicht genutzten Indikatoren zu befreien, wird die Funktion [IndicatorRelease\(\)](#) verwendet, zu deren Indikatorhandle passiert wird.

### Bemerkung

Puffernummern: 0 - MAIN\_LINE, 1 - SIGNAL\_LINE.

### Beispiel:

```
//+-----+
//|                                     Demo_iRVI.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Der Indikator zeigt, wie Daten erhalten werden sollen"
#property description "Indikator-Puffer für den technischen Indikator iRVI."
#property description "Symbol und Zeitrahmen, in denen der Indikator berechnet wird,"
#property description "sind durch die Parameter symbol und period angegeben."
#property description "Erstellungsweise des Handles ist durch den 'type' Parameter (Typ)"
#property description "Alle andere Parameter sind wie im normalen Relative Vigor Index"

#property indicator_separate_window
#property indicator_buffers 2
```

```

#property indicator_plots 2
//--- RVI bauen
#property indicator_label1 "RVI"
#property indicator_type1 DRAW_LINE
#property indicator_color1 clrGreen
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//--- Signal bauen
#property indicator_label2 "Signal"
#property indicator_type2 DRAW_LINE
#property indicator_color2 clrRed
#property indicator_style2 STYLE_SOLID
#property indicator_width2 1
//+-----+
//| Aufzählung der Erstellungsweisen des Handles |
//+-----+
enum Creation
{
    Call_iRVI,          // iRVI verwenden
    Call_IndicatorCreate // IndicatorCreate verwenden
};
//--- Eingabeparameter
input Creation      type=Call_iRVI;          // Funktionstyp
input int           ma_period=10;            // Zeitraum für die Berechnung
input string        symbol=" ";              // Symbol
input ENUM_TIMEFRAMES period=PERIOD_CURRENT; // Zeitrahmen
//--- Indicator-Puffer
double      RVIBuffer[];
double      SignalBuffer[];
//--- Eine Variable um Handle des Indikators iRVI zu speichern
int  handle;
//--- Variable für Speicherung
string name=symbol;
//--- Name des Indikators auf dem Chart
string short_name;
//--- die Anzahl der Werte im Indikator Relative Vigor Index gespeichert wird
int  bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- Bindung von Arrays zu den Indikator-Puffern
    SetIndexBuffer(0,RVIBuffer,INDICATOR_DATA);
    SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
    //--- das Symbol, auf das der Indikator gebaut wird, definieren
    name=symbol;
    //--- Leerzeichen aus dem linken und rechten löschen
    StringTrimRight(name);

```

```

StringTrimLeft(name);
//--- Wenn nach diesem die Länge des String name ist Null
if(StringLen(name)==0)
{
    //--- das Symbol aus dem Chart, auf dem der Indikator läuft, nehmen
    name=_Symbol;
};
//--- Erstellen ein Handel des Indikators
if(type==Call_iRVI)
    handle=iRVI(name,period,ma_period);
else
{
    //--- die Struktur mit Werten von Indikatorparametern ausfüllen
    MqlParam pars[1];
    //--- Zeitraum für die Berechnung
    pars[0].type=TYPE_INT;
    pars[0].integer_value=ma_period;
    handle=IndicatorCreate(name,period,IND_RVI,1,pars);
};
//--- Wenn Handle konnte nicht erstellt werden
if(handle==INVALID_HANDLE)
{
    //--- darüber schreiben und Nummer des Fehlers anzeigen
    PrintFormat("Handle des Indikators iRVI für das Paar %s/%s konnte nicht erstellt
                name,
                EnumToString(period),
                GetLastError());
    //--- Arbeit des Indikators ist früher geendet
    return(INIT_FAILED);
};
//--- Das Paar, Symbol/Zeitrahen, auf deren Relative Vigor Index berechnet war, zeige
short_name=StringFormat("iRSI(%s/%s, %d, %d)",name,EnumToString(period),ma_period);
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- normale Initialisierung des Indikators
return(INIT_SUCCEEDED);
};
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])

```

```

{
//--- Anzahl der Werte des Indikators iRVI zu kopieren
    int values_to_copy;
//--- Anzahl der berechneten Werte im Indikator finden
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() hat %d zurückgegeben, Fehlercode ist %d",calculated,GetLastError());
        return(0);
    }
//--- wenn dies der erste Start der Berechnung des Indikators ist oder die Anzahl der
//--- oder wenn Sie brauchen den Indikator für zwei oder mehr Balken (was bedeutet, dass
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
    {
        //--- wenn Array RVIBuffer größer als die Anzahl der Werte in iRVI auf symbol/period
        //--- sonst kopieren wir weniger als Größe der Indikator-Puffer
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    }
    else
    {
        //--- dann ist unser Indikator nicht das erste Mal berechnet, und seit dem letzten
        //--- nicht mehr als ein Balken für die Berechnung hinzugefügt war
        values_to_copy=(rates_total-prev_calculated)+1;
    }
//--- Die Arrays mit den Werten aus dem Indikator iRVI ausfüllen
//--- Wenn FillArrayFromBuffer false zurückgegeben hat, dann die Daten nicht bereit sind
    if(!FillArrayFromBuffer(RVIBuffer,SignalBuffer,handle,values_to_copy)) return(0);
//--- Nachricht bilden
    string comm=StringFormat("%s ==> Aktualisierte Werte im Indikator %s: %d",
                               TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                               short_name,
                               values_to_copy);
//--- Hilfsnachrichtung auf dem Chart anzeigen
    Comment(comm);
//--- die Anzahl der Werte im Indikator Relative Vigor Index speichern
    bars_calculated=calculated;
//--- den Wert prev_calculated für den nächsten Anruf zurückgeben
    return(rates_total);
}

//+-----+
//| Indikator-Puffer aus dem Indikator iRVI ausfüllen |
//+-----+
bool FillArrayFromBuffer(double &rvi_buffer[], // Indikator-Puffer der Werte von iRVI
                        double &signal_buffer[], // Indikator-Puffer der Signallinie
                        int ind_handle, // Handle des Indikators iRVI
                        int amount // Anzahl der Werte, die kopiert werden
                        )
{

```

```

//--- Fehlercode rücksetzen
ResetLastError();

//--- Teil des Arrays iRVIBuffer mit Werten auf Indikator-Puffer mit Index 0 ausfüllen
if(CopyBuffer(ind_handle,0,0,amount,rvi_buffer)<0)
{
    //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
    PrintFormat("Daten konnte nicht aus dem Indikator iRVI kopiert werden, Fehlercode: %d", GetLastError());
    //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden kann
    return(false);
};

//--- Teil des Arrays SignalBuffer mit Werten auf Indikator-Puffer mit Index 1 ausfüllen
if(CopyBuffer(ind_handle,1,0,amount,signal_buffer)<0)
{
    //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
    PrintFormat("Daten konnte nicht aus dem Indikator iRVI kopiert werden, Fehlercode: %d", GetLastError());
    //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden kann
    return(false);
};

//--- Alles gelang
return(true);
};

//+-----+
//| Indicator deinitialization function |
//+-----+

void OnDeinit(const int reason)
{
    //--- Das Chart nach der Löschung des Indikators leeren
    Comment("");
};

```

## iStdDev

Gibt das Handle des Indikators Standard Deviation zurück. Hat nur einen Puffer.

```
int iStdDev(
    string          symbol,          // Symbolname
    ENUM_TIMEFRAMES period,          // Periode
    int             ma_period,       // Mittelungsperiode
    int             ma_shift,        // horizontale Verschiebung des Indikators
    ENUM_MA_METHOD  ma_method,       // Glättungstyp
    ENUM_APPLIED_PRICE applied_price // Preistyp oder Handle
);
```

### Parameter

*symbol*

[in] Symbolname des Instrumentes, dessen Daten für Berechnung des Indikators verwendet werden. [NULL](#) bedeutet das laufende Symbol.

*period*

[in] Wert der Periode kann einer der Aufzählungswerte [ENUM\\_TIMEFRAMES](#) sein, 0 bedeutet das laufende Timeframe.

*ma\_period*

[in] Mittelungsperiode für Berechnung des Indikators.

*ma\_shift*

[in] Verschiebung des Indikators in Bezug auf das Preischart.

*ma\_method*

[in] Mittelungsmethode. Kann einer der Werte [ENUM\\_MA\\_METHOD](#) sein.

*applied\_price*

[in] Der verwendete Preis. Kann eine der Preiskonstanten [ENUM\\_APPLIED\\_PRICE](#) oder Handle eines anderen Indikators sein.

### Rückgabewert

Gibt das Handle des angegebenen technischen Indikators zurück, beim Misserfolg gibt [INVALID\\_HANDLE](#) zurück. Um Computerspeicher aus nicht genutzten Indikatoren zu befreien, wird die Funktion [IndicatorRelease\(\)](#) verwendet, zu deren Indikatorhandle passiert wird.

### Beispiel:

```
//+-----+
//|                                     Demo_iStdDev.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
```



```

#property description "Der Indikator zeigt, wie Daten erhalten werden sollen"
#property description "Indikator-Puffer für den technischen Indikator iStdDev."
#property description "Symbol und Zeitrahmen, in denen der Indikator berechnet wird,"
#property description "sind durch die Parameter symbol und period angegeben."
#property description "Erstellungsweise des Handles ist durch den 'type' Parameter (Ty
#property description "Alle andere Parameter sind wie im normalen Indikator Standard I

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots 1
//--- iStdDev bauen
#property indicator_label1 "iStdDev"
#property indicator_type1 DRAW_LINE
#property indicator_color1 clrMediumSeaGreen
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//+-----+
//| Aufzählung der Erstellungsweisen des Handles |
//+-----+
enum Creation
{
    Call_iStdDev,          // iStdDev verwenden
    Call_IndicatorCreate   // IndicatorCreate verwenden
};
//--- Eingabeparameter
input Creation          type=Call_iStdDev;          // Funktionstyp
input int               ma_period=20;              // Mittelungszeitraum
input int               ma_shift=0;                // Verschiebung
input ENUM_MA_METHOD    ma_method=MODE_SMA;        // Art der Glättung
input ENUM_APPLIED_PRICE applied_price=PRICE_CLOSE; // Preistyp
input string            symbol=" ";                // Symbol
input ENUM_TIMEFRAMES   period=PERIOD_CURRENT;    // Zeitrahmen
//--- Indikator-Puffer
double          iStdDevBuffer[];
//--- Eine Variable um Handle des Indikators iStdDev zu speichern
int    handle;
//--- Variable für Speicherung
string name=symbol;
//--- Name des Indikators auf dem Chart
string short_name;
//--- die Anzahl der Werte im Indikator Standard Deviation gespeichert wird
int    bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- Bindung von Array zum Indikator-Puffer
    SetIndexBuffer(0,iStdDevBuffer,INDICATOR_DATA);

```

```

//--- Verschiebung definieren
    PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- das Symbol, auf das der Indikator gebaut wird, definieren
    name=symbol;
//--- Leerzeichen aus dem linken und rechten löschen
    StringTrimRight(name);
    StringTrimLeft(name);
//--- Wenn nach diesem die Länge des String name ist Null
    if(StringLen(name)==0)
    {
        //--- das Symbol aus dem Chart, auf dem der Indikator läuft, nehmen
        name=_Symbol;
    }
//--- Erstellen ein Handel des Indikators
    if(type==Call_iStdDev)
        handle=iStdDev(name,period,ma_period,ma_shift,ma_method,applied_price);
    else
    {
        //--- die Struktur mit Werten von Indikatorparametern ausfüllen
        MqlParam pars[4];
        //--- Periode
        pars[0].type=TYPE_INT;
        pars[0].integer_value=ma_period;
        //--- Verschiebung
        pars[1].type=TYPE_INT;
        pars[1].integer_value=ma_shift;
        //--- Art der Glättung
        pars[2].type=TYPE_INT;
        pars[2].integer_value=ma_method;
        //--- Preistyp
        pars[3].type=TYPE_INT;
        pars[3].integer_value=applied_price;
        handle=IndicatorCreate(name,period,IND_STDDEV,4,pars);
    }
//--- Wenn Handle konnte nicht erstellt werden
    if(handle==INVALID_HANDLE)
    {
        //--- darüber schreiben und Nummer des Fehlers anzeigen
        PrintFormat("Handle des Indikators iStdDev für das Paar %s/%s konnte nicht erstellt werden",
            name,
            EnumToString(period),
            GetLastError());
        //--- Arbeit des Indikators ist früher geendet
        return(INIT_FAILED);
    }
//--- Das Paar, Symbol/Zeitraumen, auf deren Standard Deviation berechnet war, zeigen
    short_name=StringFormat("iStdDev(%s/%s, %d, %d, %s, %s)",name,EnumToString(period),
        ma_period,ma_shift,EnumToString(ma_method),EnumToString(applied_price));
    IndicatorSetString(INDICATOR_SHORTNAME,short_name);

```

```

//--- normale Initialisierung des Indikators
    return(INIT_SUCCEEDED);
; }
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{
//--- Anzahl der Werte des Indikators iStdDev zu kopieren
    int values_to_copy;
//--- Anzahl der berechneten Werte im Indikator finden
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() hat %d zurückgegeben, Fehlercode ist %d",calculated,GetLastError());
        return(0);
    }
; }
//--- wenn dies der erste Start der Berechnung des Indikators ist oder die Anzahl der
//--- oder wenn Sie brauchen den Indikator für zwei oder mehr Balken (was bedeutet, dass
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
    {
        //--- wenn Array iStdDevBuffer größer als die Anzahl der Werte in iStdDev auf system
        //--- sonst kopieren wir weniger als Größe der Indikator-Puffer
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    }
; }
else
{
    //--- dann ist unser Indikator nicht das erste Mal berechnet, und seit dem letzten
    //--- nicht mehr als ein Balken für die Berechnung hinzugefügt war
    values_to_copy=(rates_total-prev_calculated)+1;
; }

//--- Das Array mit den Werten aus dem Indikator Standard Deviation ausfüllen
//--- Wenn FillArrayFromBuffer false zurückgegeben hat, dann die Daten nicht bereit sind
    if(!FillArrayFromBuffer(iStdDevBuffer,ma_shift,handle,values_to_copy)) return(0);
//--- Nachricht bilden
    string comm=StringFormat("%s ==> Aktualisierte Werte im Indikator %s: %d",
                              TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                              short_name,
                              values_to_copy);

```

```

//--- Hilfsnachrichtung auf dem Chart anzeigen
    Comment(comm);
//--- die Anzahl der Werte im Indikator Standard Deviation speichern
    bars_calculated=calculated;
//--- den Wert prev_calculated für den nächsten Anruf zurückgeben
    return(rates_total);
; }

//+-----+
//| Indikator-Puffer aus dem Indikator iStdDev ausfüllen |
//+-----+
bool FillArrayFromBuffer(double &std_buffer[], // Indikator-Puffer der Linie von Start
                        int std_shift,         // Verschiebung der Linie Standard Deviation
                        int ind_handle,        // Handle des Indikators iStdDev
                        int amount            // Anzahl der Werte, die kopiert werden
                        )
{
//--- Fehlercode rücksetzen
    ResetLastError();
//--- Teil des Arrays iStdDevBuffer mit Werten auf Indikator-Puffer mit Index 0 ausfüllen
    if(CopyBuffer(ind_handle,0,-std_shift,amount,std_buffer)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iStdDev kopiert werden, Fehlercode: %d", GetLastError());
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden konnte
        return(false);
    }
; }

//--- Alles gelang
    return(true);
; }

//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
//--- Das Chart nach der Löschung des Indikators leeren
    Comment("");
; }

```

## iStochastic

Gibt das Handle des Indikators Stochastic Oscillator zurück.

```
int iStochastic(
    string          symbol,          // Symbolname
    ENUM_TIMEFRAMES period,          // Periode
    int             Kperiod,          // K-Periode (Anzahl der Bars für Berechnung)
    int             Dperiod,          // D-Periode (Periode der ersten Glättung)
    int             slowing,          // abschliessende Glättung
    ENUM_MA_METHOD  ma_method,        // Glättungstyp
    ENUM_STO_PRICE  price_field       // Berechnungsmethode der Stochastic
);
```

### Parameter

*symbol*

[in] Symbolname des Instrumentes, dessen Daten für Berechnung des Indikators verwendet werden. [NULL](#) bedeutet das laufende Symbol.

*period*

[in] Wert der Periode kann einer der Aufzählungswerte [ENUM\\_TIMEFRAMES](#), 0 bedeutet das laufende Timeframe.

*Kperiod*

[in] K-Periode(Anzahl der Bars) für Berechnung der Linie %K.

*Dperiod*

[in] Mittelungsperiode für Berechnung der Linie %D.

*slowing*

[in] Wert von Verlangsamung.

*ma\_method*

[in] Mittelungsmethode. Kann einer der Aufzählungswerte [ENUM\\_MA\\_METHOD](#) sein.

*price\_field*

[in] Parameter der Preiswahl für Berechnung. Kann einer Werte [ENUM\\_STO\\_PRICE](#) sein.

### Rückgabewert

Gibt das Handle des angegebenen technischen Indikators zurück, beim Misserfolg gibt [INVALID\\_HANDLE](#) zurück. Um Computerspeicher aus nicht genutzten Indikatoren zu befreien, wird die Funktion [IndicatorRelease\(\)](#) verwendet, zu deren Indikatorhandle passiert wird.

### Bemerkung

Puffernummern: 0 - MAIN\_LINE, 1 - SIGNAL\_LINE.

### Beispiel:

```
//+-----+
//|                                     Demo_iStochastic.mq5 |
```

```

//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Der Indikator zeigt, wie Daten erhalten werden sollen"
#property description "Indikator-Puffer für den technischen Indikator iStochastic."
#property description "Symbol und Zeitrahmen, in denen der Indikator berechnet wird,"
#property description "sind durch die Parameter symbol und period angegeben."
#property description "Erstellungsweise des Handles ist durch den 'type' Parameter (Ty
#property description "Alle andere Parameter sind wie im normalen Indikator Stochastic

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots 2
//--- Stochastic bauen
#property indicator_label1 "Stochastic"
#property indicator_type1  DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- Signal bauen
#property indicator_label2 "Signal"
#property indicator_type2  DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- die Grenzwerte des Indikators definieren
#property indicator_minimum 0
#property indicator_maximum 100
//--- Horizontale Ebenen im Indikatorsfenster
#property indicator_level1  -100.0
#property indicator_level2  100.0
//+-----+
//| Aufzählung der Erstellungsweisen des Handles |
//+-----+
enum Creation
{
    Call_iStochastic,      // iStochastic verwenden
    Call_IndicatorCreate    // IndicatorCreate verwenden
};
//--- Eingabeparameter
input Creation      type=Call_iStochastic;    // Funktionstyp
input int           Kperiod=5;                // K-Periode (Anzahl der Balken
input int           Dperiod=3;                // D-Periode (Periode der primär
input int           slowing=3;                // Periode der endgültigen Glätt
input ENUM_MA_METHOD ma_method=MODE_SMA;      // Art der Glättung
input ENUM_STO_PRICE price_field=STO_LOWHIGH; // Methode zur Berechnung von St

```

```

input string          symbol=" ";           // Symbol
input ENUM_TIMEFRAMES period=PERIOD_CURRENT; // Zeitrahmen
//--- Indicator-Puffer
double              StochasticBuffer[];
double              SignalBuffer[];
//--- Eine Variable um Handle des Indikators iStochastic zu speichern
int                handle;
//--- Variable für Speicherung
string name=symbol;
//--- Name des Indikators auf dem Chart
string short_name;
//--- die Anzahl der Werte im Indikator Stochastic Oscillator gespeichert wird
int                bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
//--- Bindung von Arrays zu den Indikator-Puffern
    SetIndexBuffer(0,StochasticBuffer,INDICATOR_DATA);
    SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
//--- das Symbol, auf das der Indikator gebaut wird, definieren
    name=symbol;
//--- Leerzeichen aus dem linken und rechten löschen
    StringTrimRight(name);
    StringTrimLeft(name);
//--- Wenn nach diesem die Länge des String name ist Null
    if(StringLen(name)==0)
    {
        //--- das Symbol aus dem Chart, auf dem der Indikator läuft, nehmen
        name=_Symbol;
    }
//--- Erstellen ein Handel des Indikators
    if(type==Call_iStochastic)
        handle=iStochastic(name,period,Kperiod,Dperiod,slowing,ma_method,price_field);
    else
    {
        //--- die Struktur mit Werten von Indikatorparametern ausfüllen
        MqlParam pars[5];
        //--- K-Periode für die Berechnung
        pars[0].type=TYPE_INT;
        pars[0].integer_value=Kperiod;
        //--- D-Periode für primäre Glättung
        pars[1].type=TYPE_INT;
        pars[1].integer_value=Dperiod;
        //--- K-Periode für finale Glättung
        pars[2].type=TYPE_INT;
        pars[2].integer_value=slowing;
        //--- Art der Glättung

```

```

    pars[3].type=TYPE_INT;
    pars[3].integer_value=ma_method;
    //--- Methode zur Berechnung von Stochastic
    pars[4].type=TYPE_INT;
    pars[4].integer_value=price_field;
    handle=IndicatorCreate(name,period,IND_STOCHASTIC,5,pars);

; }

//--- Wenn Handle konnte nicht erstellt werden
if(handle==INVALID_HANDLE)
{
    //--- darüber schreiben und Nummer des Fehlers anzeigen
    PrintFormat("Handle des Indikators iStochastic für das Paar %s/%s konnte nicht e
                name,
                EnumToString(period),
                GetLastError());

    //--- Arbeit des Indikators ist früher geendet
    return(INIT_FAILED);

; }

//--- Das Paar, Symbol/Zeitraumen, auf deren Stochastic Oscillator berechnet war, zeig
short_name=StringFormat("iStochastic(%s/%s, %d, %d, %d, %s, %s)",name,EnumToString
                Kperiod,Dperiod,slowing,EnumToString(ma_method),EnumToStrin
    IndicatorSetString(INDICATOR_SHORTNAME,short_name);

//--- normale Initialisierung des Indikators
return(INIT_SUCCEEDED);

; }

//+-----+
//| Custom indicator iteration function |
//+-----+

int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])

{
    //--- Anzahl der Werte des Indikators iStochastic zu kopieren
    int values_to_copy;
    //--- Anzahl der berechneten Werte im Indikator finden
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() hat %d zurückgegeben, Fehlercode ist %d",calculate
        return(0);

    ; }

    //--- wenn dies der erste Start der Berechnung des Indikators ist oder die Anzahl der

```



```

//--- oder wenn Sie brauchen den Indikator für zwei oder mehr Balken (was bedeutet, da
if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
{
    //--- wenn Array StochasticBuffer größer als die Anzahl der Werte in iStochastic
    //--- sonst kopieren wir weniger als Größe der Indikator-Puffer
    if(calculated>rates_total) values_to_copy=rates_total;
    else
        values_to_copy=calculated;
};
else
{
    //--- dann ist unser Indikator nicht das erste Mal berechnet, und seit dem letzten
    //--- nicht mehr als ein Balken für die Berechnung hinzugefügt war
    values_to_copy=(rates_total-prev_calculated)+1;
};
//--- Die Arrays mit den Werten aus dem Indikator iStochastic ausfüllen
//--- Wenn FillArraysFromBuffers false zurückgegeben hat, dann die Daten nicht bereit
if(!FillArraysFromBuffers(StochasticBuffer,SignalBuffer,handle,values_to_copy)) return;
//--- Nachricht bilden
string comm=StringFormat("%s ==> Aktualisierte Werte im Indikator %s: %d",
                        TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                        short_name,
                        values_to_copy);
//--- Hilfsnachrichtung auf dem Chart anzeigen
Comment(comm);
//--- die Anzahl der Werte im Indikator Stochastic Oscillator speichern
bars_calculated=calculated;
//--- den Wert prev_calculated für den nächsten Anruf zurückgeben
return(rates_total);
};
//+-----+
//| Indikator-Puffer aus dem Indikator iStochastic ausfüllen |
//+-----+
bool FillArraysFromBuffers(double &main_buffer[], // Indikator-Puffer der Werte vor
                        double &signal_buffer[], // Indikator-Puffer der Signallin
                        int ind_handle, // Handle des Indikators iStochastic
                        int amount // Anzahl der Werte, die kopiert
                        )
{
    //--- Fehlercode rücksetzen
    ResetLastError();
    //--- Teil des Arrays StochasticBuffer mit Werten auf Indikator-Puffer mit Index 0 ausfüllen
    if(CopyBuffer(ind_handle,MAIN_LINE,0,amount,main_buffer)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iStochastic kopiert werden, Fehlercode: %d",
                    GetLastError());
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet wurde
        return(false);
    };
    //--- Teil des Arrays SignalBuffer mit Werten auf Indikator-Puffer mit Index 1 ausfüllen

```

```

    if(CopyBuffer(ind_handle,SIGNAL_LINE,0,amount,signal_buffer)<0)
    {
        //-- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iStochastic kopiert werden, Fehlercode: %d", GetLastError());
        //-- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden konnte
        return(false);
    }
    //-- Alles gelang
    return(true);
};

//+-----+
//| Indicator deinitialization function |
//+-----+

void OnDeinit(const int reason)
{
    //-- Das Chart nach der Löschung des Indikators leeren
    Comment("");
};

```

## iTEMA

Gibt das Handle des Indikators Triple Exponential Moving Average zurück. Hat nur einen Puffer.

```
int iTEMA(
    string          symbol,          // Symbolname
    ENUM_TIMEFRAMES period,          // Periode
    int             ma_period,        // Mittelungsperiode
    int             ma_shift,         // horizontale Verschiebung des Indikators
    ENUM_APPLIED_PRICE applied_price // Preistyp oder Handle
);
```

### Parameter

*symbol*

[in] Symbolname des Instrumentes, dessen Daten für Berechnung des Indikators verwendet werden. [NULL](#) bedeutet das laufende Symbol.

*period*

[in] Wert der Periode kann einer der Aufzählungswerte [ENUM\\_TIMEFRAMES](#) sein, 0 bedeutet das laufende Timeframe.

*ma\_period*

[in] Periode(Anzahl der Bars) für Berechnung des Indikators.

*ma\_shift*

[in] Verschiebung des Indikators in Bezug auf das Preischart.

*applied\_price*

[in] Der verwendete Preis. Kann einer der Preiskonstanten [ENUM\\_APPLIED\\_PRICE](#) oder Handle eines anderen Indikators sein.

### Rückgabewert

Gibt das Handle des angegebenen technischen Indikators zurück, beim Misserfolg gibt [INVALID\\_HANDLE](#) zurück. Um Computerspeicher aus nicht genutzten Indikatoren zu befreien, wird die Funktion [IndicatorRelease\(\)](#) verwendet, zu deren Indikatorhandle passiert wird.

### Beispiel:

```
//+-----+
//|                                     Demo_iTEMA.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Der Indikator zeigt, wie Daten erhalten werden sollen"
#property description "Indikator-Puffer für den technischen Indikator iTEMA."
#property description "Symbol und Zeitrahmen, in denen der Indikator berechnet wird,"
#property description "sind durch die Parameter symbol und period angegeben."
```

```

#property description "Erstellungsweise des Handles ist durch den 'type' Parameter (Ty
#property description "Alle andere Parameter sind wie im normalen Indikator Triple Exp

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots 1
//--- iTEMA bauen
#property indicator_label1 "iTEMA"
#property indicator_type1 DRAW_LINE
#property indicator_color1 clrRed
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//+-----+
//| Aufzählung der Erstellungsweisen des Handles |
//+-----+
enum Creation
{
    Call_iTEMA,          // iTEMA verwenden
    Call_IndicatorCreate // IndicatorCreate verwenden
};
//--- Eingabeparameter
input Creation      type=Call_iTEMA;          // Funktionstyp
input int           ma_period=14;              // Mittelungszeitraum
input int           ma_shift=0;                // Verschiebung
input ENUM_APPLIED_PRICE applied_price=PRICE_CLOSE; // Preistyp
input string        symbol=" ";               // Symbol
input ENUM_TIMEFRAMES period=PERIOD_CURRENT;  // Zeitrahmen
//--- Indikator-Puffer
double            iTEMABuffer[];
//--- Eine Variable um Handle des Indikators iTEMA zu speichern
int    handle;
//--- Variable für Speicherung
string name=symbol;
//--- Name des Indikators auf dem Chart
string short_name;
//--- die Anzahl der Werte im Indikator Triple Exponential Moving Average gespeichert
int    bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- Bindung von Array zum Indikator-Puffer
    SetIndexBuffer(0,iTEMABuffer,INDICATOR_DATA);
    //--- Verschiebung definieren
    PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
    //--- das Symbol, auf das der Indikator gebaut wird, definieren
    name=symbol;
    //--- Leerzeichen aus dem linken und rechten löschen

```

```

StringTrimRight(name);
StringTrimLeft(name);
//--- Wenn nach diesem die Länge des String name ist Null
if(StringLen(name)==0)
{
    //--- das Symbol aus dem Chart, auf dem der Indikator läuft, nehmen
    name=_Symbol;
};
//--- Erstellen ein Handel des Indikators
if(type==Call_iTEMA)
    handle=iTEMA(name,period,ma_period,ma_shift,applied_price);
else
{
    //--- die Struktur mit Werten von Indikatorparametern ausfüllen
    MqlParam pars[3];
    //--- Periode
    pars[0].type=TYPE_INT;
    pars[0].integer_value=ma_period;
    //--- Verschiebung
    pars[1].type=TYPE_INT;
    pars[1].integer_value=ma_shift;
    //--- Preistyp
    pars[2].type=TYPE_INT;
    pars[2].integer_value=applied_price;
    handle=IndicatorCreate(name,period,IND_TEMA,3,pars);
};
//--- Wenn Handle konnte nicht erstellt werden
if(handle==INVALID_HANDLE)
{
    //--- darüber schreiben und Nummer des Fehlers anzeigen
    PrintFormat("Handle des Indikators iTEMA für das Paar %s/%s konnte nicht erstellt
                name,
                EnumToString(period),
                GetLastError());
    //--- Arbeit des Indikators ist früher geendet
    return(INIT_FAILED);
};
//--- Das Paar, Symbol/Zeitraumen, auf deren Triple Exponential Moving Average berechne
short_name=StringFormat("iTEMA(%s/%s, %d, %d, %s)",name,EnumToString(period),
                        ma_period,ma_shift,EnumToString(applied_price));
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- normale Initialisierung des Indikators
return(INIT_SUCCEEDED);
};
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,

```

```

        const datetime &time[],
        const double &open[],
        const double &high[],
        const double &low[],
        const double &close[],
        const long &tick_volume[],
        const long &volume[],
        const int &spread[])
    {
        //--- Anzahl der Werte des Indikators iTEMA zu kopieren
        int values_to_copy;
        //--- Anzahl der berechneten Werte im Indikator finden
        int calculated=BarsCalculated(handle);
        if(calculated<=0)
        {
            PrintFormat("BarsCalculated() hat %d zurückgegeben, Fehlercode ist %d",calculated,GetLastError());
            return(0);
        }
        //--- wenn dies der erste Start der Berechnung des Indikators ist oder die Anzahl der
        //--- oder wenn Sie brauchen den Indikator für zwei oder mehr Balken (was bedeutet, dass
        if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
        {
            //--- wenn Array iTEMABuffer größer als die Anzahl der Werte in iTEMA auf symbol
            //--- sonst kopieren wir weniger als Größe der Indikator-Puffer
            if(calculated>rates_total) values_to_copy=rates_total;
            else
                values_to_copy=calculated;
        }
        else
        {
            //--- dann ist unser Indikator nicht das erste Mal berechnet, und seit dem letzten
            //--- nicht mehr als ein Balken für die Berechnung hinzugefügt war
            values_to_copy=(rates_total-prev_calculated)+1;
        }
        //--- Das Array mit den Werten aus dem Indikator Triple Exponential Moving Average aus
        //--- Wenn FillArrayFromBuffer false zurückgegeben hat, dann die Daten nicht bereit sind
        if(!FillArrayFromBuffer(iTEMABuffer,ma_shift,handle,values_to_copy)) return(0);
        //--- Nachricht bilden
        string comm=StringFormat("%s ==> Aktualisierte Werte im Indikator %s: %d",
                                   TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                                   short_name,
                                   values_to_copy);
        //--- Hilfsnachrichtung auf dem Chart anzeigen
        Comment(comm);
        //--- die Anzahl der Werte im Indikator Triple Exponential Moving Average speichern
        bars_calculated=calculated;
        //--- den Wert prev_calculated für den nächsten Anruf zurückgeben
        return(rates_total);
    }
    //+-----+

```

```

//| Indikator-Puffer aus dem Indikator iTEMA ausfüllen |
//+-----+
bool FillArrayFromBuffer(double &tema_buffer[], // Indikator-Puffer der Werte von Trips
                        int t_shift,           // Verschiebung der Linie
                        int ind_handle,        // Handle des Indikators iTEMA
                        int amount             // Anzahl der Werte, die kopiert werden
                        )
{
    //--- Fehlercode rücksetzen
    ResetLastError();
    //--- Teil des Arrays iTEMABuffer mit Werten auf Indikator-Puffer mit Index 0 ausfüllen
    if(CopyBuffer(ind_handle,0,-t_shift,amount,tema_buffer)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iTEMA kopiert werden, Fehlercode: %d", GetLastError());
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden konnte
        return(false);
    }
    //--- Alles gelang
    return(true);
}
//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
    //--- Das Chart nach der Löschung des Indikators leeren
    Comment("");
}

```

## iTriX

Gibt das Handle des Indikators Triple Exponential Moving Averages Oscillator zurück. Hat nur einen Puffer.

```
int iTriX(
    string          symbol,          // Symbolname
    ENUM_TIMEFRAMES period,          // Periode
    int             ma_period,        // Mittelungsperiode
    ENUM_APPLIED_PRICE applied_price  // Preistyp oder Handle
);
```

### Parameter

*symbol*

[in] Symbolname des Instrumentes, dessen Daten für Berechnung des Indikators verwendet werden. [NULL](#) bedeutet das laufende Symbol.

*period*

[in] Wert der Periode kann einer der Aufzählungswerte [ENUM\\_TIMEFRAMES](#) sein, sein 0 bedeutet das laufende Timeframe.

*ma\_period*

[in] Periode(Anzahl der Bars) für Berechnung des Indikators.

*applied\_price*

[in] Der verwendete Preis. Kann eine der Preiskonstanten [ENUM\\_APPLIED\\_PRICE](#) oder Handle eines anderen Indikators sein.

### Rückgabewert

Gibt das Handle des angegebenen technischen Indikators zurück, beim Misserfolg gibt [INVALID\\_HANDLE](#) zurück. Um Computerspeicher aus nicht genutzten Indikatoren zu befreien, wird die Funktion [IndicatorRelease\(\)](#) verwendet, zu deren Indikatorhandle passiert wird.

### Beispiel:

```
//+-----+
//|                                     Demo_iTriX.mq5 |
//|               Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Der Indikator zeigt, wie Daten erhalten werden sollen"
#property description "Indikator-Puffer für den technischen Indikator iTriX."
#property description "Symbol und Zeitrahmen, in denen der Indikator berechnet wird,"
#property description "sind durch die Parameter symbol und period angegeben."
#property description "Erstellungsweise des Handles ist durch den 'type' Parameter (T

#property indicator_separate_window
```



```

#property indicator_buffers 1
#property indicator_plots 1
//--- iTriX bauen
#property indicator_label1 "iTriX"
#property indicator_type1 DRAW_LINE
#property indicator_color1 clrRed
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//+-----+
//| Aufzählung der Erstellungsweisen des Handles |
//+-----+
enum Creation
{
    Call_iTriX,          // iTriX verwenden
    Call_IndicatorCreate // IndicatorCreate verwenden
};
//--- Eingabeparameter
input Creation      type=Call_iTriX;          // Funktionstyp
input int           ma_period=14;             // Periode
input ENUM_APPLIED_PRICE applied_price=PRICE_CLOSE; // Preistyp
input string        symbol=" ";              // Symbol
input ENUM_TIMEFRAMES period=PERIOD_CURRENT; // Zeitrahmen
//--- Indicator-Puffer
double iTriXBuffer[];
//--- Eine Variable um Handle des Indikators iTriX zu speichern
int handle;
//--- Variable für Speicherung
string name=symbol;
//--- Name des Indikators auf dem Chart
string short_name;
//--- die Anzahl der Werte im Indikator Triple Exponential Moving Averages Oscillator
int bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- Bindung von Array zum Indikator-Puffer
    SetIndexBuffer(0,iTriXBuffer,INDICATOR_DATA);
    //--- das Symbol, auf das der Indikator gebaut wird, definieren
    name=symbol;
    //--- Leerzeichen aus dem linken und rechten löschen
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- Wenn nach diesem die Länge des String name ist Null
    if(StringLen(name)==0)
    {
        //--- das Symbol aus dem Chart, auf dem der Indikator läuft, nehmen
        name=_Symbol;
    }
}

```

```

    ;}
//--- Erstellen ein Handel des Indikators
if(type==Call_iTriX)
    handle=iTriX(name,period,ma_period,applied_price);
else
{
    //--- die Struktur mit Werten von Indikatorparametern ausfüllen
    MqlParam pars[2];
    //--- Periode
    pars[0].type=TYPE_INT;
    pars[0].integer_value=ma_period;
    //--- Preistyp
    pars[1].type=TYPE_INT;
    pars[1].integer_value=applied_price;
    handle=IndicatorCreate(name,period,IND_TRIX,2,pars);
}
//--- Wenn Handle konnte nicht erstellt werden
if(handle==INVALID_HANDLE)
{
    //--- darüber schreiben und Nummer des Fehlers anzeigen
    PrintFormat("Handle des Indikators iTriX für das Paar %s/%s konnte nicht erstellt
                name,
                EnumToString(period),
                GetLastError());
    //--- Arbeit des Indikators ist früher geendet
    return(INIT_FAILED);
}
//--- Das Paar, Symbol/Zeitraumen, auf deren Triple Exponential Moving Averages Oscill
short_name=StringFormat("iTriX(%s/%s, %d, %s)",name,EnumToString(period),
                        ma_period,EnumToString(applied_price));
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- normale Initialisierung des Indikators
return(INIT_SUCCEEDED);
;}
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{
    //--- Anzahl der Werte des Indikators iTriX zu kopieren

```

```

    int values_to_copy;
    //--- Anzahl der berechneten Werte im Indikator finden
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() hat %d zurückgegeben, Fehlercode ist %d",calculated,GetLastError());
        return(0);
    }
    //--- wenn dies der erste Start der Berechnung des Indikators ist oder die Anzahl der
    //--- oder wenn Sie brauchen den Indikator für zwei oder mehr Balken (was bedeutet, dass
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
    {
        //--- wenn Array iTriXBuffer größer als die Anzahl der Werte in iTriX auf symbol
        //--- sonst kopieren wir weniger als Größe der Indikator-Puffer
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    }
    else
    {
        //--- dann ist unser Indikator nicht das erste Mal berechnet, und seit dem letzten
        //--- nicht mehr als ein Balken für die Berechnung hinzugefügt war
        values_to_copy=(rates_total-prev_calculated)+1;
    }
    //--- Das Array mit den Werten aus dem Indikator Triple Exponential Moving Averages Oscillator
    //--- Wenn FillArrayFromBuffer false zurückgegeben hat, dann die Daten nicht bereit sind
    if(!FillArrayFromBuffer(iTriXBuffer,handle,values_to_copy)) return(0);
    //--- Nachricht bilden
    string comm=StringFormat("%s ==> Aktualisierte Werte im Indikator %s: %d",
                              TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                              short_name,
                              values_to_copy);
    //--- Hilfsnachrichtung auf dem Chart anzeigen
    Comment(comm);
    //--- die Anzahl der Werte im Indikator Triple Exponential Moving Averages Oscillator
    bars_calculated=calculated;
    //--- den Wert prev_calculated für den nächsten Anruf zurückgeben
    return(rates_total);
}

//+-----+
//| Indikator-Puffer aus dem Indikator iTriX ausfüllen |
//+-----+
bool FillArrayFromBuffer(double &trix_buffer[], // Indikator-Puffer der Werte von TriX
                        int ind_handle,         // Handle des Indikators iTriX
                        int amount              // Anzahl der Werte, die kopiert werden
                        )
{
    //--- Fehlercode rücksetzen
    ResetLastError();
    //--- Teil des Arrays iTriXBuffer mit Werten auf Indikator-Puffer mit Index 0 ausfüllen

```

```

    if(CopyBuffer(ind_handle,0,0,amount,trix_buffer)<0)
    {
        //-- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iTriX kopiert werden, Fehlercode: %d", GetLastError());
        //-- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden kann
        return(false);
    }
    ;}

/-- Alles gelang
return(true);
;}

//+-----+
//| Indicator deinitialization function |
//+-----+

void OnDeinit(const int reason)
{
    //-- Das Chart nach der Löschung des Indikators leeren
    Comment("");
    ;}

```

## iWPR

Gibt das Handle des Indikators Larry Williams' Percent Range zurück. Hat nur einen Puffer.

```
int iWPR(
    string      symbol,           // Symbolname
    ENUM_TIMEFRAMES period,       // Periode
    int         calc_period       // Mittelungsperiode
);
```

### Parameter

*symbol*

[in] Symbolname des Instrumentes, dessen Daten für Berechnung des Indikators verwendet werden. NULL bedeutet das laufende Symbol.

*period*

[in] Wert der Periode kann einer der Aufzählungswerte ENUM\_TIMEFRAMES sein, 0 bedeutet das laufende Timeframe.

*calc\_period*

[in] Periode(Anzahl der Bars) für Berechnung des Indikators.

### Rückgabewert

Gibt das Handle des angegebenen technischen Indikators zurück, beim Misserfolg gibt INVALID\_HANDLE zurück. Um Computerspeicher aus nicht genutzten Indikatoren zu befreien, wird die Funktion IndicatorRelease() verwendet, zu deren Indikatorhandle passiert wird.

### Beispiel:

```
//+-----+
//|                                     Demo_iWPR.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Der Indikator zeigt, wie Daten erhalten werden sollen"
#property description "Indikator-Puffer für den technischen Indikator iWPR."
#property description "Symbol und Zeitrahmen, in denen der Indikator berechnet wird,"
#property description "sind durch die Parameter symbol und period angegeben."
#property description "Erstellungsweise des Handles ist durch den 'type' Parameter (T)"

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- iWPR bauen
#property indicator_label1   "iWPR"
#property indicator_type1    DRAW_LINE
#property indicator_color1   clrCyan
```

```

#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//--- die Grenzwerte des Indikators definieren
#property indicator_minimum -100
#property indicator_maximum 0
//--- Horizontale Ebenen im Indikatorsfenster
#property indicator_level1 -20.0
#property indicator_level2 -80.0
//+-----+
//| Aufzählung der Erstellungsweisen des Handles |
//+-----+
enum Creation
{
    Call_iWPR,          // iWPR verwenden
    Call_IndicatorCreate // IndicatorCreate verwenden
};
//--- Eingabeparameter
input Creation      type=Call_iWPR;          // Funktionstyp
input int           calc_period=14;          // Periode
input string        symbol=" ";              // Symbol
input ENUM_TIMEFRAMES period=PERIOD_CURRENT; // Zeitrahmen
//--- Indikator-Puffer
double iWPRBuffer[];
//--- Eine Variable um Handle des Indikators iWPR zu speichern
int handle;
//--- Variable für Speicherung
string name=symbol;
//--- Name des Indikators auf dem Chart
string short_name;
//--- die Anzahl der Werte im Indikator Larry Williams' Percent Range gespeichert wird
int bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- Bindung von Array zum Indikator-Puffer
    SetIndexBuffer(0,iWPRBuffer,INDICATOR_DATA);
    //--- das Symbol, auf das der Indikator gebaut wird, definieren
    name=symbol;
    //--- Leerzeichen aus dem linken und rechten löschen
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- Wenn nach diesem die Länge des String name ist Null
    if(StringLen(name)==0)
    {
        //--- das Symbol aus dem Chart, auf dem der Indikator läuft, nehmen
        name=_Symbol;
    }
};

```

```

//--- Erstellen ein Handel des Indikators
if(type==Call_iWPR)
    handle=iWPR(name,period,calc_period);
else
{
    //--- die Struktur mit Werten von Indikatorparametern ausfüllen
    MqlParam pars[1];
    //--- Periode
    pars[0].type=TYPE_INT;
    pars[0].integer_value=calc_period;
    handle=IndicatorCreate(name,period,IND_WPR,1,pars);
};

//--- Wenn Handle konnte nicht erstellt werden
if(handle==INVALID_HANDLE)
{
    //--- darüber schreiben und Nummer des Fehlers anzeigen
    PrintFormat("Handle des Indikators iWPR für das Paar %s/%s konnte nicht erstellt
                name,
                EnumToString(period),
                GetLastError());

    //--- Arbeit des Indikators ist früher geendet
    return(INIT_FAILED);
};

//--- Das Paar, Symbol/Zeitraumen, auf deren Larry Williams' Percent Range berechnet v
short_name=StringFormat("iWPR(%s/%s, %d)",name,EnumToString(period),calc_period);
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- normale Initialisierung des Indikators
return(INIT_SUCCEEDED);
};

//+-----+
//| Custom indicator iteration function |
//+-----+

int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{
    //--- Anzahl der Werte des Indikators iWPR zu kopieren
    int values_to_copy;
    //--- Anzahl der berechneten Werte im Indikator finden
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {

```

```

        PrintFormat("BarsCalculated() hat %d zurückgegeben, Fehlercode ist %d",calculate
        return(0);
    };
}

//--- wenn dies der erste Start der Berechnung des Indikators ist oder die Anzahl der
//--- oder wenn Sie brauchen den Indikator für zwei oder mehr Balken (was bedeutet, da
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
    {
        //--- wenn Array iWPRBuffer größer als die Anzahl der Werte in iWPR auf symbol/p
        //--- sonst kopieren wir weniger als Größe der Indikator-Puffer
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    };
else
    {
        //--- dann ist unser Indikator nicht das erste Mal berechnet, und seit dem letzte
        //--- nicht mehr als ein Balken für die Berechnung hinzugefügt war
        values_to_copy=(rates_total-prev_calculated)+1;
    };

//--- Das Array mit den Werten aus dem Indikator Larry Williams' Percent Range ausfüll
//--- Wenn FillArrayFromBuffer false zurückgegeben hat, dann die Daten nicht bereit s
    if(!FillArrayFromBuffer(iWPRBuffer,handle,values_to_copy)) return(0);
//--- Nachricht bilden
    string comm=StringFormat("%s ==> Aktualisierte Werte im Indikator %s: %d",
                                TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                                short_name,
                                values_to_copy);
//--- Hilfsnachrichtung auf dem Chart anzeigen
    Comment(comm);
//--- die Anzahl der Werte im Indikator Larry Williams' Percent Range speichern
    bars_calculated=calculated;
//--- den Wert prev_calculated für den nächsten Anruf zurückgeben
    return(rates_total);
};

//+-----+
//| Indikator-Puffer aus dem Indikator iWPR ausfüllen |
//+-----+
bool FillArrayFromBuffer(double &wpr_buffer[], // Indikator-Puffer der Werte von Will
                        int ind_handle,        // Handle des Indikators iWPR
                        int amount             // Anzahl der Werte, die kopiert werden
                        )
{
    //--- Fehlercode rücksetzen
    ResetLastError();
    //--- Teil des Arrays iWPRBuffer mit Werten auf Indikator-Puffer mit Index 0 ausfüllen
    if(CopyBuffer(ind_handle,0,0,amount,wpr_buffer)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iWPR kopiert werden, Fehlercod
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berech

```



```
        return(false);
    ;}

//--- Alles gelang
    return(true);
; }

//+-----+
//| Indicator deinitialization function |
//+-----+

void OnDeinit(const int reason)
{
//--- Das Chart nach der Löschung des Indikators leeren
    Comment("");
; }
```

## iVIDyA

Gibt das Handle des Indikators Variable Index Dynamic Average zurück. Hat nur einen Puffer.

```
int iVIDyA(
    string          symbol,          // Symbolname
    ENUM_TIMEFRAMES period,          // Periode
    int             cmo_period,      // Periode Chande Momentum
    int             ema_period,      // EMA Glättungsperiode
    int             ma_shift,        // horizontale Verschiebung des Indikators
    ENUM_APPLIED_PRICE applied_price // Preistyp oder Handle
);
```

### Parameter

*symbol*

[in] Symbolname des Instrumentes, dessen Daten für Berechnung des Indikators verwendet werden. [NULL](#) bedeutet das laufende Symbol.

*period*

[in] Wert der Periode kann einer der Aufzählungswerte [ENUM\\_TIMEFRAMES](#) sein, 0 bedeutet das laufende Timeframe.

*cmo\_period*

[in] Periode(Anzahl der Bars) für Berechnung Chande Momentum Oscillator.

*ema\_period*

[in] Periode(Anzahl der Bars) EMA für Berechnung des Glättungsfaktors.

*ma\_shift*

[in] Verschiebung des Indikators in Bezug auf das Preischart.

*applied\_price*

[in] Der verwendete Preis. Kann eine der Aufzählungskonstanten [ENUM\\_APPLIED\\_PRICE](#) oder handle eines anderen Indikators sein.

### Rückgabewert

Gibt das Handle des angegebenen technischen Indikators zurück, beim Misserfolg gibt [INVALID\\_HANDLE](#) zurück. Um Computerspeicher aus nicht genutzten Indikatoren zu befreien, wird die Funktion [IndicatorRelease\(\)](#) verwendet, zu deren Indikatorhandle passiert wird.

### Beispiel:

```
//+-----+
//|                                     Demo_iVIDyA.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

```

#property description "Der Indikator zeigt, wie Daten erhalten werden sollen"
#property description "Indikator-Puffer für den technischen Indikator iVIDyA."
#property description "Symbol und Zeitrahmen, in denen der Indikator berechnet wird,"
#property description "sind durch die Parameter symbol und period angegeben."
#property description "Erstellungsweise des Handles ist durch den 'type' Parameter (Ty
#property description "Alle andere Parameter sind wie im normalen Indikator Variable 1

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots 1
//--- iVIDyA bauen
#property indicator_label1 "iVIDyA"
#property indicator_type1 DRAW_LINE
#property indicator_color1 clrBlue
#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//+-----+
//| Aufzählung der Erstellungsweisen des Handles |
//+-----+
enum Creation
{
    Call_iVIDyA,          // iVIDyA verwenden
    Call_IndicatorCreate  // IndicatorCreate verwenden
};
//--- Eingabeparameter
input Creation          type=Call_iVIDyA;          // Funktionstyp
input int               cmo_period=15;            // Periode Chande Momentum
input int               ema_period=12;            // Zeitraum von Glättungsfaktor
input int               ma_shift=0;               // Verschiebung
input ENUM_APPLIED_PRICE applied_price=PRICE_CLOSE; // Preistyp
input string            symbol=" ";               // Symbol
input ENUM_TIMEFRAMES   period=PERIOD_CURRENT;    // Zeitrahmen
//--- Indikator-Puffer
double                iVIDyABuffer[];
//--- Eine Variable um Handle des Indikators iVIDyA zu speichern
int    handle;
//--- Variable für Speicherung
string name=symbol;
//--- Name des Indikators auf dem Chart
string short_name;
//--- die Anzahl der Werte im Indikator Variable Index Dynamic Average gespeichert wi
int    bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- Bindung von Array zum Indikator-Puffer
    SetIndexBuffer(0,iVIDyABuffer,INDICATOR_DATA);

```

```

//--- Verschiebung definieren
    PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- das Symbol, auf das der Indikator gebaut wird, definieren
    name=symbol;
//--- Leerzeichen aus dem linken und rechten löschen
    StringTrimRight(name);
    StringTrimLeft(name);
//--- Wenn nach diesem die Länge des String name ist Null
    if(StringLen(name)==0)
    {
        //--- das Symbol aus dem Chart, auf dem der Indikator läuft, nehmen
        name=_Symbol;
    }
//--- Erstellen ein Handel des Indikators
    if(type==Call_iVIDyA)
        handle=iVIDyA(name,period,cmo_period,ema_period,ma_shift,applied_price);
    else
    {
        //--- die Struktur mit Werten von Indikatorparametern ausfüllen
        MqlParam pars[4];
        //--- Periode Chande Momentum
        pars[0].type=TYPE_INT;
        pars[0].integer_value=cmo_period;
        //--- Zeitraum von Glättungsfaktor
        pars[1].type=TYPE_INT;
        pars[1].integer_value=ema_period;
        //--- Verschiebung
        pars[2].type=TYPE_INT;
        pars[2].integer_value=ma_shift;
        //--- Preistyp
        pars[3].type=TYPE_INT;
        pars[3].integer_value=applied_price;
        handle=IndicatorCreate(name,period,IND_VIDYA,4,pars);
    }
//--- Wenn Handle konnte nicht erstellt werden
    if(handle==INVALID_HANDLE)
    {
        //--- darüber schreiben und Nummer des Fehlers anzeigen
        PrintFormat("Handle des Indikators iVIDyA für das Paar %s/%s konnte nicht erstellt werden",
            name,
            EnumToString(period),
            GetLastError());
        //--- Arbeit des Indikators ist früher geendet
        return(INIT_FAILED);
    }
//--- Das Paar, Symbol/Zeitrahen, auf deren Triple Exponential Moving Average berechnet
    short_name=StringFormat("iVIDyA(%s/%s, %d, %d, %d, %s)",name,EnumToString(period),
        cmo_period,ema_period,ma_shift,EnumToString(applied_price));
    IndicatorSetString(INDICATOR_SHORTNAME,short_name);

```

```

//--- normale Initialisierung des Indikators
    return(INIT_SUCCEEDED);
; }
//+-----+
//| Custom indicator iteration function |
//+-----+

int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{
//--- Anzahl der Werte des Indikators iVIDyA zu kopieren
    int values_to_copy;
//--- Anzahl der berechneten Werte im Indikator finden
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() hat %d zurückgegeben, Fehlercode ist %d",calculated,GetLastError());
        return(0);
    }
; }
//--- wenn dies der erste Start der Berechnung des Indikators ist oder die Anzahl der
//--- oder wenn Sie brauchen den Indikator für zwei oder mehr Balken (was bedeutet, dass
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
    {
        //--- wenn Array iVIDyABuffer größer als die Anzahl der Werte in iVIDyA auf symbol
        //--- sonst kopieren wir weniger als Größe der Indikator-Puffer
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    }
; }
else
{
    //--- dann ist unser Indikator nicht das erste Mal berechnet, und seit dem letzten
    //--- nicht mehr als ein Balken für die Berechnung hinzugefügt war
    values_to_copy=(rates_total-prev_calculated)+1;
; }

//--- Das Array mit den Werten aus dem Indikator Variable Index Dynamic Average ausfüllen
//--- Wenn FillArrayFromBuffer false zurückgegeben hat, dann die Daten nicht bereit sind
    if(!FillArrayFromBuffer(iVIDyABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- Nachricht bilden
    string comm=StringFormat("%s ==> Aktualisierte Werte im Indikator %s: %d",
                              TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                              short_name,
                              values_to_copy);

```

```

//--- Hilfsnachrichtung auf dem Chart anzeigen
    Comment(comm);
//--- die Anzahl der Werte im Indikator Variable Index Dynamic Average speichern
    bars_calculated=calculated;
//--- den Wert prev_calculated für den nächsten Anruf zurückgeben
    return(rates_total);
; }

//+-----+
//| Indikator-Puffer aus dem Indikator iVIDyA ausfüllen |
//+-----+
bool FillArrayFromBuffer(double &vidya_buffer[], // Indikator-Puffer der Werte von Vari
                        int v_shift,           // Verschiebung der Linie
                        int ind_handle,        // Handle des Indikators iVIDyA
                        int amount            // Anzahl der Werte, die kopiert werden
                        )
{
//--- Fehlercode rücksetzen
    ResetLastError();
//--- Teil des Arrays iVIDyABuffer mit Werten auf Indikator-Puffer mit Index 0 ausfüllen
    if(CopyBuffer(ind_handle,0,-v_shift,amount,vidya_buffer)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iVIDyA kopiert werden, Fehlercode: %d", GetLastError());
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden konnte
        return(false);
    }
//--- Alles gelang
    return(true);
; }

//+-----+
//| Indicator deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
//--- Das Chart nach der Löschung des Indikators leeren
    Comment("");
; }

```

## iVolumes

Gibt das Handle des Indikators der Volumen zurück. Hat nur einen Puffer.

```
int iVolumes(
    string          symbol,          // Symbolname
    ENUM_TIMEFRAMES period,          // Periode
    ENUM_APPLIED_VOLUME applied_volume // Volumentyp
)
```

### Parameter

*symbol*

[in] Symbolname des Instrumentes, dessen Daten für Berechnung des Indikators verwendet werden. NULL bedeutet das laufende Symbol.

*period*

[in] Wert der Periode kann einer der Aufzählungswerte [ENUM\\_TIMEFRAMES](#) sein, 0 bedeutet das laufende Timeframe.

*applied\_volume*

[in] Das verwendete Volumen. Kann einer der Aufzählungswerte [ENUM\\_APPLIED\\_VOLUME](#) sein.

### Rückgabewert

Gibt das Handle des angegebenen technischen Indikators zurück, beim Misserfolg gibt [INVALID\\_HANDLE](#) zurück. Um Computerspeicher aus nicht genutzten Indikatoren zu befreien, wird die Funktion [IndicatorRelease\(\)](#) verwendet, zu deren Indikatorhandle passiert wird.

### Beispiel:

```
//+-----+
//|                                     Demo_iVolumes.mq5 |
//|                                     Copyright 2011, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+

#property copyright "Copyright 2011, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Der Indikator zeigt, wie Daten erhalten werden sollen"
#property description "Indikator-Puffer für den technischen Indikator iVolumes."
#property description "Symbol und Zeitrahmen, in denen der Indikator berechnet wird,"
#property description "sind durch die Parameter symbol und period angegeben."
#property description "Erstellungsweise des Handles ist durch den 'type' Parameter (T)"

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots 1
//--- iVolumes bauen
#property indicator_label1 "iVolumes"
#property indicator_type1  DRAW_COLOR_HISTOGRAM
#property indicator_color1 clrGreen, clrRed
```

```

#property indicator_style1 STYLE_SOLID
#property indicator_width1 1
//+-----+
//| Aufzählung der Erstellungsweisen des Handles |
//+-----+
enum Creation
{
    Call_iVolumes,          // iVolumes verwenden
    Call_IndicatorCreate    // IndicatorCreate verwenden
};
//--- Eingabeparameter
input Creation      type=Call_iVolumes;          // Funktionstyp
input ENUM_APPLIED_VOLUME applied_volume=VOLUME_TICK; // Volumentyp
input string        symbol=" ";                  // Symbol
input ENUM_TIMEFRAMES period=PERIOD_CURRENT;    // Zeitrahmen
//--- Indicator-Puffer
double      iVolumesBuffer[];
double      iVolumesColors[];
//--- Eine Variable um Handle des Indikators iVolumes zu speichern
int  handle;
//--- Variable für Speicherung
string name=symbol;
//--- Name des Indikators auf dem Chart
string short_name;
//--- die Anzahl der Werte im Indikator Volumes gespeichert wird
int  bars_calculated=0;
//+-----+
//| Custom indicator initialization function |
//+-----+
int OnInit()
{
    //--- Bindung von Array zu den Indikator-Puffern
    SetIndexBuffer(0,iVolumesBuffer,INDICATOR_DATA);
    SetIndexBuffer(1,iVolumesColors,INDICATOR_COLOR_INDEX);
    //--- das Symbol, auf das der Indikator gebaut wird, definieren
    name=symbol;
    //--- Leerzeichen aus dem linken und rechten löschen
    StringTrimRight(name);
    StringTrimLeft(name);
    //--- Wenn nach diesem die Länge des String name ist Null
    if(StringLen(name)==0)
    {
        //--- das Symbol aus dem Chart, auf dem der Indikator läuft, nehmen
        name=_Symbol;
    };
    //--- Erstellen ein Handel des Indikators
    if(type==Call_iVolumes)
        handle=iVolumes(name,period,applied_volume);
    else

```



```

{
    //--- die Struktur mit Werten von Indikatorparametern ausfüllen
    MqlParam pars[1];
    //--- Preistyp
    pars[0].type=TYPE_INT;
    pars[0].integer_value=applied_volume;
    handle=IndicatorCreate(name,period,IND_VOLUMES,1,pars);
};
//--- Wenn Handle konnte nicht erstellt werden
if(handle==INVALID_HANDLE)
{
    //--- darüber schreiben und Nummer des Fehlers anzeigen
    PrintFormat("Handle des Indikators iVolumes für das Paar %s/%s konnte nicht erst
                name,
                EnumToString(period),
                GetLastError());
    //--- Arbeit des Indikators ist früher geendet
    return(INIT_FAILED);
};
//--- Das Paar von Symbol/Zeitraumen, auf deren Volumes berechnet war, anzeigen
short_name=StringFormat("iVolumes(%s/%s, %s)",name,EnumToString(period),EnumToStrin
IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- normale Initialisierung des Indikators
return(INIT_SUCCEEDED);
};
//+-----+
//| Custom indicator iteration function |
//+-----+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
{
    //--- Anzahl der Werte des Indikators iVolumes zu kopieren
    int values_to_copy;
    //--- Anzahl der berechneten Werte im Indikator finden
    int calculated=BarsCalculated(handle);
    if(calculated<=0)
    {
        PrintFormat("BarsCalculated() hat %d zurückgegeben, Fehlercode ist %d",calculate
        return(0);
    };
    //--- wenn dies der erste Start der Berechnung des Indikators ist oder die Anzahl der

```

```

//--- oder wenn Sie brauchen den Indikator für zwei oder mehr Balken (was bedeutet, da
    if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated)
    {
        //--- wenn Array iVolumesBuffer größer als die Anzahl der Werte in iVolumes auf
        //--- sonst kopieren wir weniger als Größe der Indikator-Puffer
        if(calculated>rates_total) values_to_copy=rates_total;
        else
            values_to_copy=calculated;
    }
else
{
    //--- dann ist unser Indikator nicht das erste Mal berechnet, und seit dem letzten
    //--- nicht mehr als ein Balken für die Berechnung hinzugefügt war
    values_to_copy=(rates_total-prev_calculated)+1;
}

//--- Die Arrays mit den Werten aus dem Indikator iVolumes ausfüllen
//--- Wenn FillArraysFromBuffers false zurückgegeben hat, dann die Daten nicht bereit
    if(!FillArraysFromBuffers(iVolumesBuffer,iVolumesColors,handle,values_to_copy)) return false;
//--- Nachricht bilden
    string comm=StringFormat("%s ==> Aktualisierte Werte im Indikator %s: %d",
                                TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                                short_name,
                                values_to_copy);
//--- Hilfsnachrichtung auf dem Chart anzeigen
    Comment(comm);
//--- die Anzahl der Werte im Indikator Volumes speichern
    bars_calculated=calculated;
//--- den Wert prev_calculated für den nächsten Anruf zurückgeben
    return(rates_total);
}

//+-----+
//| Indikator-Puffer aus dem Indikator iVolumes ausfüllen |
//+-----+
bool FillArraysFromBuffers(double &volume_buffer[], // Indikator-Puffer der Werte von iVolumes
                           double &color_buffer[], // Farbpuffer des Indikators
                           int ind_handle, // Handle des Indikators iVolumes
                           int amount // Anzahl der Werte, die kopiert werden
                           )
{
    //--- Fehlercode rücksetzen
    ResetLastError();
    //--- Teil des Arrays iVolumesBuffer mit Werten auf Indikator-Puffer mit Index 0 ausfüllen
    if(CopyBuffer(ind_handle,0,0,amount,volume_buffer)<0)
    {
        //--- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iVolumes kopiert werden, Fehlercode: %d",GetLastError());
        //--- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden konnte
        return(false);
    }
}

//--- Teil des Arrays iVolumesColors mit Werten auf Indikator-Puffer mit Index 1 ausfüllen

```

```

    if(CopyBuffer(ind_handle,1,0,amount,color_buffer)<0)
    {
        //-- wenn die Kopie fehlschlägt, Fehlercode anzeigen
        PrintFormat("Daten konnte nicht aus dem Indikator iVolumes kopiert werden, Fehlercode: %d", GetLastError());
        //-- Beenden mit Null-Ergebnis - dies bedeutet, dass der Indikator nicht berechnet werden kann
        return(false);
    }
    ;}

/-- Alles gelang
return(true);
; }

//+-----+
//| Indicator deinitialization function |
//+-----+

void OnDeinit(const int reason)
{
    //-- Das Chart nach der Löschung des Indikators leeren
    Comment("");
; }

```

## Arbeit mit Ergebnisse der Optimierung

Funktionen für die Organisation benutzerdefinierte Verarbeitung der Ergebnisse der Optimierung im Strategie-Tester. Sie können während der Optimierung in Test-Agenten sowie lokal in Expert Advisors und Skripte aufgerufen werden.

Wenn Sie einen Expert Advisor im Strategie-Tester laufen, können Sie Ihre eigenen Daten-Array auf den einfachen Typen oder [einfachen Strukturen](#) (sie enthalten keine Strings, Objekte der Klasse oder Objekte den dynamischen Arrays) erstellen. Dieser Datensatz kann mit Hilfe der Funktion [FrameAdd\(\)](#) in einer speziellen Struktur, die ein Frame genannt wird, gespeichert werden. Bei der Optimierung eines Expert Advisor kann jeder Agent eine Reihe von Frames in das Terminal senden. Alle empfangenen Frames werden in der Datei \*.MQD im Ordner terminal\_directory/MQL5/Files/Tester genannt als Expert Advisor geschrieben. Sie werden in der Reihenfolge wie sie von den Agenten empfangen werden geschrieben. Empfang eines Frames im Kundenterminal dem Testagent erstellt ein Ereignis [TesterPass](#).

Frames können im Speicher des Computers und in einer Datei mit dem angegebenen Namen gespeichert werden. Die MQL5 Sprache setzt keine Einschränkungen für die Anzahl der Frames.

| Funktion                          | Aktion                                                                                                                                                                                    |
|-----------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">FrameFirst</a>        | Verschiebt einen Zeiger der Lesung des Frames auf den Anfang und rückt den zuvor angegebenen Filter                                                                                       |
| <a href="#">FrameFilter</a>       | Setzt den Filter der Lesung des Frames und verschiebt den Zeiger auf den Anfang                                                                                                           |
| <a href="#">FrameNext</a>         | Liest einen Frame und verschiebt den Zeiger auf das nächste                                                                                                                               |
| <a href="#">FrameInputs</a>       | Empfängt <a href="#">Input-Parameter</a> , auf die der Frame gebildet ist                                                                                                                 |
| <a href="#">FrameAdd</a>          | Fügt einen Frame mit Daten hinzu                                                                                                                                                          |
| <a href="#">ParameterGetRange</a> | Empfängt die Information über den Bereich der Werte und den Schritt der Veränderung für <a href="#">input-Variable</a> bei der Optimierung eines Expert Advisors in Strategie-Tester      |
| <a href="#">ParameterSetRange</a> | Erstellt Regeln für die Verwendung von <a href="#">input-Variable</a> bei der Optimierung eines Expert Advisors in Strategie-Tester: Wert, Schritt der Veränderung, Anfangs- und Endwerte |

Sehen Sie auch

[Teststatistik](#), [Information über das ausgeführte MQL5-Programm](#)

## FrameFirst

Verschiebt einen Zeiger der Lesung des Frames auf den Anfang und rückt den angegebenen Filter.

```
bool FrameFirst();
```

### Rückgabewert

Gibt bei Erfolg true zurück, ansonsten false. Um Informationen über den Fehler zu erhalten, rufen Sie die Funktion [GetLastError\(\)](#).

## FrameFilter

Setzt den Filter der Lesung des Frames und verschiebt den Zeiger auf den Anfang.

```
bool FrameFilter(  
    const string name,          // Öffentlicher Name/Label  
    long id                    // Öffentliche ID  
);
```

### Rückgabewert

Gibt bei Erfolg true zurück, ansonsten false. Um Informationen über den Fehler zu erhalten, rufen Sie die Funktion [GetLastError\(\)](#).

### Bemerkung

Wenn ein leerer String als ersten Parameter übergeben wird, der Filter funktioniert nur mit einem numerischen Parameter, dh nur Frames mit der angegebenen id angezeigt werden. Wenn der Wert des zweiten Parameters ist [ULONG\\_MAX](#), nur Text Filter arbeitet.

Aufruf von [FrameFilter\("", ULONG\\_MAX\)](#) ist mit dem Aufruf von [FrameFirst\(\)](#) gleichbedeutend, das entspricht der Nutzung keiner Filter.

## FrameNext

Liest einen Frame und verschiebt den Zeiger auf das nächste. Es gibt zwei Varianten der Funktion.

### 1. Aufruf, um einen numerischen Wert zu erhalten

```
bool FrameNext(  
    ulong& pass,      // Nummer des Durchgangs in der Optimierung, auf dem der Frame  
    string& name,     // Öffentlicher Name/Label  
    long& id,         // Öffentliche ID  
    double& value     // Wert  
);
```

### 2. Aufruf, um alle Daten des Frames zu erhalten

```
bool FrameNext(  
    ulong& pass,      // Nummer des Durchgangs in der Optimierung, auf dem der Frame  
    string& name,     // Öffentlicher Name/Label  
    long& id,         // Öffentliche ID  
    double& value,    // Wert  
    void& data[]      // Array eines beliebigen Typs  
);
```

#### Parameter

*pass*

[out] Nummer des Durchgangs in der Optimierung im Strategie-Tester.

*name*

[out] Der Name des Bezeichners.

*id*

[out] Der Wert des Bezeichners.

*value*

[out] Ein einzelner numerischer Wert.

*data*

[out] Ein Array von beliebigen Typs.

#### Rückgabewert

Gibt bei Erfolg true zurück, ansonsten false. Um Informationen über den Fehler zu erhalten, rufen Sie die Funktion [GetLastError\(\)](#).

#### Bemerkung

In der zweiten Version des Aufrufs, müssen Sie die empfangenen Daten im Array *data[]* korrekt verarbeiten.

## FrameInputs

Empfängt [Input-Parameter](#), auf die der Frame mit dem angegebenen Durchgangsnummer gebildet ist.

```
bool FrameInputs(  
    ulong    pass,                // Nummer des Durchgangs in der Optimierung  
    string&  parameters[],        // Array von String wie "parameterN=valueN"  
    uint&    parameters_count    // Gesamtzahl der Parameter  
);
```

### Parameter

*pass*

[out] Nummer des Durchgangs in der Optimierung im Strategie-Tester.

*parameters*

[out] Ein String-Array mit der Beschreibung der Namen und Parameterwerten.

*parameters\_count*

[out] Die Anzahl der Elemente im Array *parameters[]*.

### Rückgabewert

Gibt bei Erfolg true zurück, ansonsten false. Um Informationen über den Fehler zu erhalten, rufen Sie die Funktion [GetLastError\(\)](#).

### Bemerkung

Nach Erhalt der Anzahl der Strings *parameters\_count* im Array *parameters[]*, können Sie eine Schleife organisieren, um durch alle Datensätze zu durchlaufen. Diese helfen Ihnen dabei, die Werte der Eingabeparameter eines Expert Advisors für die angegebene Nummer des Durchgangs zu erhalten.



## FrameAdd

Fügt einen Frame mit Daten hinzu. Es gibt zwei Varianten der Funktion.

### 1. Hinzufügen von Daten aus einer Datei

```
bool FrameAdd(  
    const string name,      // Öffentlicher Name/Label  
    long id,               // Öffentliche ID  
    double value,          // Wert  
    const string filename  // Name der Datendatei  
);
```

### 2. Hinzufügen von Daten aus einem Array eines beliebigen Typs

```
bool FrameAdd(  
    const string name,      // Öffentlicher Name/Label  
    long id,               // Öffentliche ID  
    double value,          // Wert  
    const void& data[]     // Array eines beliebigen Typs  
);
```

#### Parameter

*name*

[in] Öffentliches Label des Frames. Es kann für Filterung in der [FrameFilter\(\)](#)-Funktion verwendet werden.

*id*

[in] Ein öffentlicher Identifikator des Frames. Es kann für Filterung in der [FrameFilter\(\)](#)-Funktion verwendet werden.

*value*

[in] Ein numerischer Wert für Schreibung im Frame. Es wird verwendet, um ein einzelnes Ergebnis des Durchlaufs wie in der [OnTester\(\)](#)-Funktion zu übertragen.

*filename*

[in] Der Name der Datei, die Daten für Hinzufügung in den Frame enthält. Die Datei muss im Ordner MQL5/Files sein.

*data*

[in] Ein Array des beliebigen Typs für Schreibung in den Frame. Wird als Referenz übergeben.

#### Rückgabewert

Gibt bei Erfolg true zurück, ansonsten false. Um Informationen über den Fehler zu erhalten, rufen Sie die Funktion [GetLastError\(\)](#).

## ParameterGetRange

Empfängt die Information über den Bereich der Werte und den Schritt der Veränderung für [input-Variable](#) bei der Optimierung eines Expert Advisors in Strategie-Tester. Es gibt zwei Varianten der Funktion.

### 1. Empfang von Information für den input-Parameter des Integer-Typs

```
bool ParameterGetRange (
    const string  name,           // Name des (input-Variable) Parameters
    bool&         enable,        // Optimierung des Parameters ist erlaubt
    long&         value,         // Wert des Parameters
    long&         start,         // Anfangswert
    long&         step,          // Schritt der Veränderung
    long&         stop           // Endwert
);
```

### 2. Empfang von Information für den input-Parameter des tatsächlichen Typs

```
bool ParameterGetRange (
    const string  name,           // Name des (input-Variable) Parameters
    double&       enable,        // Optimierung des Parameters ist erlaubt
    double&       value,         // Wert des Parameters
    double&       start,         // Anfangswert
    double&       step,          // Schritt der Veränderung
    double&       stop           // Endwert
);
```

### Parameter

*name*

[in] Identifikator der [input-Variable](#). Diese Variablen sind externe Parameter des Programms, deren Werte man beim Start auf einem Chart oder während eines einzelnen Testens angeben kann.

*enable*

[out] Merkmal, dass dieser Parameter für Durchsuchen der Werte bei der Optimierung in Strategie-Tester verwendet werden kann.

*value*

[out] Wert des Parameters.

*start*

[out] Anfangswert des Parameters bei der Optimierung.

*step*

[out] Schritt der Veränderung des Parameters beim Durchsuchen seiner Werte.

*stop*

[out] Endwert des Parameters bei der Optimierung.

### Rückgabewert

Gibt bei Erfolg true zurück, ansonsten false. Um Informationen über den Fehler zu erhalten, verwenden Sie die Funktion [GetLastError\(\)](#).

#### Bemerkung

Die Funktion kann nur aus den Handlern [OnTesterInit\(\)](#), [OnTesterPass\(\)](#) und [OnTesterDeinit\(\)](#) aufgerufen werden. Sie ist für den Erhalt von Wert und Veränderungsbereich der Eingabeparameter des Expert Advisors bei der Optimierung in Strategie-Tester bestimmt.

Beim Aufruf in [OnTesterInit\(\)](#) kann die erhaltene Information für Neudefinierung der Regeln für Durchsuchen jeder [input-Variable](#) mit Hilfe der Funktion [ParameterSetRange\(\)](#) verwendet werden. So kann man die neuen Werte Start, Stop, Schritt einstellen und sogar diesen Parameter von der Optimierung unabhängig von den Einstellungen in Strategie-Tester völlig ausschließen. Dies ermöglicht es Ihnen eigene Skripten zu erstellen, um den Raum von Eingabeparametern bei der Optimierung zu verwalten, das heißt von der Optimierung einige Parameter je nach den Werten der wichtigsten Parameter des Expert Advisors auszuschließen.

#### Beispiel:

```

#property description "Expert Advisor demonstriert die Funktion ParameterGetRange()."
#property description "Man muss in Strategie-Tester im Optimierung-Modus starten"
//--- input parameters
input int          Input1=1;
input double       Input2=2.0;
input bool         Input3=false;
input ENUM_DAY_OF_WEEK Input4=SUNDAY;

//+-----+
//| Expert initialization function |
//+-----+
int OnInit()
{
    //--- Expert Advisor ist nur für die Arbeit in Strategie-Tester bestimmt
    if (!MQL5InfoInteger(MQL5_OPTIMIZATION))
    {
        MessageBox("Man muss in Strategie-Tester im Optimierung-Modus starten!");
        //--- beenden wir die Arbeit des Expert Advisors vorfristig und entfernen wir a
        return(INIT_FAILED);
    }
    //--- erfolgreicher Abschluss der Initialisierung
    return(INIT_SUCCEEDED);
}
//+-----+
//| TesterInit function |
//+-----+
void OnTesterInit()
{
    //--- Beispiel für den input-Parameter des Typs long
    string name="Input1";
    bool enable;
    long par1,par1_start,par1_step,par1_stop;
    ParameterGetRange(name,enable,par1,par1_start,par1_step,par1_stop);
    Print("Der erste Parameter");
    PrintFormat("%s=%d enable=%s from %d to %d with step=%d",
                name,par1,(string)enable,par1_start,par1_stop,par1_step);
    //--- Beispiel für den input-Parameter des Typs double
    name="Input2";
    double par2,par2_start,par2_step,par2_stop;
    ParameterGetRange(name,enable,par2,par2_start,par2_step,par2_stop);
    Print("Der zweite Parameter");
    PrintFormat("%s=%G enable=%s from %G to %G with step=%G",
                name,par2,(string)enable,par2_start,par2_stop,par2_step);

    //--- Beispiel für den input-Parameter des Typs bool
    name="Input3";
    long par3,par3_start,par3_step,par3_stop;
    ParameterGetRange(name,enable,par3,par3_start,par3_step,par3_stop);
    Print("Der dritte Parameter");
    PrintFormat("%s=%s enable=%s from %s to %s",
                name,(string)par3,(string)enable,
                (string)par3_start,(string)par3_stop);
    //--- Beispiel für den input-Parameter des Aufzählungstyps
    name="Input4";
    long par4,par4_start,par4_step,par4_stop;
    ParameterGetRange(name,enable,par4,par4_start,par4_step,par4_stop);
    Print("Der vierte Parameter");
    PrintFormat("%s=%s enable=%s from %s to %s",
                name,EnumToString((ENUM_DAY_OF_WEEK)par4),(string)enable,
                EnumToString((ENUM_DAY_OF_WEEK)par4_start),
                EnumToString((ENUM_DAY_OF_WEEK)par4_stop));
}

```

```
    }  
    //+-----+  
    //| TesterDeinit function |  
    //+-----+  
    void OnTesterDeinit()  
    {  
        //--- diese Meldung wird nach dem Abschluss der Optimierung ausgegeben  
        Print(__FUNCTION__, " Optimisation completed");  
    }
```

## ParameterSetRange

Erstellt Regeln für die Verwendung von [input-Variable](#) bei der Optimierung eines Expert Advisors in Strategie-Tester: Wert, Schritt der Veränderung, Anfangs-und Endwerte. Es gibt zwei Varianten der Funktion.

### 1. Einstellung von Werten für den input-Parameter des Integer-Typs

```
bool ParameterSetRange (
    const string  name,           // Name des (input-Variable) Parameters
    bool          enable,        // Optimierung des Parameters erlauben
    long          value,         // Wert des Parameters
    long          start,         // Anfangswert
    long          step,          // Schritt der Veränderung
    long          stop           // Endwert
);
```

### 2. Einstellung von Werten für den input-Parameter des tatsächlichen Typs

```
bool ParameterSetRange (
    const string  name,           // Name des (input-Variable) Parameters
    double        enable,        // Optimierung des Parameters erlauben
    double        value,         // Wert des Parameters
    double        start,         // Anfangswert
    double        step,          // Schritt der Veränderung
    double        stop           // Endwert
);
```

#### Parameter

*name*

[in] Identifikator der [input](#) oder [sinput](#) Variable. Diese Variablen sind externe Parameter des Programms, deren Werte man beim Start angeben kann.

*enable*

[In] Aktivieren Sie diesen Parameter für Durchsuchen der Werte bei der Optimierung in Strategie-Tester.

*value*

[in] Wert des Parameters.

*start*

[in] Anfangswert des Parameters bei der Optimierung.

*step*

[in] Schritt der Veränderung des Parameters beim Durchsuchen seiner Werte.

*stop*

[in] Endwert des Parameters bei der Optimierung.

#### Rückgabewert

Gibt bei Erfolg true zurück, ansonsten false. Um Informationen über den Fehler zu erhalten, verwenden Sie die Funktion [GetLastError\(\)](#).

#### Bemerkung

Die Funktion kann nur aus dem Handler [OnTesterInit\(\)](#) beim Start der Optimierung im Strategie-Tester aufgerufen werden. Sie ist für die Einstellung von Bereich und Schritt der Veränderung des Parameters bestimmt, und erlaubt diesen Parameter von der Optimierung, unabhängig von den Einstellungen in Strategie-Tester, völlig auszuschließen. Auch erlaubt sie in der Optimierung sogar die mit dem Modifikator sinput erklärten Variablen zu verwenden.

Die Funktion `ParameterSetRange()` ermöglicht es Ihnen eigene Skripten der Optimierung des Expert-Advisors in Strategie-Tester je nach den Werten seinen wichtigsten Parameter zu erstellen, das heißt von der Optimierung die notwendigen Eingabeparameter ein- oder auszuschließen, und auch die notwendigen Bereich und Schritt der Veränderung einzustellen.

## Arbeit mit Ereignissen

Funktionen für die Arbeit mit Benutzerereignissen und Ereignissen des Timers . Ausser dieser Funktionen gibt es auch Sonderfunktionen für die Verarbeitung [Vorwegparameter](#).

| Funktion                                 | Massnahme                                                                                                                     |
|------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">EventSetMillisecondTimer</a> | Startet den Generator der hochauflösenden Ereignissetimer mit einer Periode von weniger als 1 Sekunde für den aktuellen Chart |
| <a href="#">EventSetTimer</a>            | Läuft Generierer der Timerereignisse mit dem angegebenen Abruf für den laufenden Chart ab                                     |
| <a href="#">EventKillTimer</a>           | Setzt auf dem laufenden Chart die Zeitbergenerierung von Ereignissen still                                                    |
| <a href="#">EventChartCustom</a>         | Generiert das Benutzerereignis für den angegebenen Chart.                                                                     |

Sehen Sie auch

[Typen der Chartereignisse](#)



## EventSetMillisecondTimer

Informiert den Terminal, dass für diesen Expert oder Indikator [Timer](#) Ereignisse mit Periodizität kleiner als eine Sekunde generiert werden sollten.

```
bool EventSetMillisecondTimer(  
    int milliseconds // Die Anzahl der Millisekunden  
);
```

### Optionen

*milliseconds*

[in] Die Anzahl der Millisekunden, die die Häufigkeit der Timer-Ereignisse definiert.

### Rückgabewert

Gibt bei Erfolg true zurück, ansonsten false. Um [Fehlercode](#) zu erhalten, rufen Sie die Funktion [GetLastError\(\)](#).

### Hinweis

Diese Funktion ist für die diejenigen Fälle verwendet, wenn Sie hochauflösende Timer brauchen, d.h. wenn Sie Timer-Ereignisse häufiger als einmal pro Sekunde zu empfangen brauchen. Wenn Sie einen einfachen Timer mit Periode von mehr als 1 Sekunde brauchen, dann verwenden Sie [EventSetTimer\(\)](#).

Im Strategie-Tester wird der Mindestabstand von 1000 Millisekunden verwendet. Im Allgemeinen erhöht das kurze Intervall Testierungszeit, weil es erhöht die Anzahl der Anrufe von Timerereignis-Handler. In Echtzeitmodus werden Timer-Ereignisse nicht mehr als 1 Mal in 10-16 Millisekunden aufgrund von Hardware-Einschränkungen generiert.

Normalerweise muss diese Funktion von die Funktion [OnInit\(\)](#) oder in [Klassenkonstruktor](#) aufgerufen werden. Um Ereignisse aus der Timer zu behandeln, muss ein Experten oder Indikator die Funktion [OnTimer\(\)](#) haben.

Jeder Expert Advisor und jeder Indikator arbeitet mit seinem Timer und empfängt Ereignisse nur von ihm. Nach Abschluss des MQL5-Programs wird der Timer zerstört, wenn es erstellt wurde, aber wurde nicht ausschalten durch die Funktion [EventKillTimer\(\)](#).

Für ein Programm kann nicht mehr als ein Timer gestartet werden. Jedes MQL5-Programm und jeder Chart hat seine eigenen Ereignis-Warteschlange, wo alle neu eingehenden Ereignisse geraten. Ist die Warteschlange bereits hat ein [Timer](#)-Ereignis oder dieses Ereignis ist in Bearbeitungslage, wird das neue Timer-Ereignis in Warteschlange vom MQL5-Programm nicht platziert.

## EventSetTimer

Gibt dem Kundenterminal an, dass Ereignisse für diesen Expert oder Indikator von [Timer](#) mit der angegebenen Periodizität generiert werden müssen.

```
bool EventSetTimer(  
    int seconds // Anzahl der Sekunden  
);
```

### Parameter

*seconds*

[in] Anzahl der Sekunden, die Periodizität des Vorkommens der Ereignisse vom Timer bestimmt.

### Rückgabewert

Im Erfolgsfall gibt true zurück, anderenfalls false. Für die Erhaltung des Kodes des [Fehlers](#) muss die Funktion [GetLastError\(\)](#) aufgerufen werden.

### Bemerkung

Normalerweise muss diese Funktion aus der Funktion [OnInit\(\)](#) oder im [Konstruktor](#) der Klasse aufgerufen werden. Für Verarbeitung der Ereignisse vom Timer muss der Expert oder der Indikator die Funktion [OnTimer\(\)](#) haben.

Jeder Expert Advisor und jeder Indikator arbeitet nur mit seinem Timer und bekommt Ereignisse nur von ihm. Bei der Beendigung des mql5-Programms wird Timer zwangsläufig vernichtet, falls er erzeugt wurde aber nicht durch die Funktion [EventKillTimer\(\)](#) ausgeschaltet wurde.

Für jedes Programm kann nicht mehr als ein Timer ablaufen lassen. Jedes MQL5-Programm und jeder Chart hat seinen eigenen Warteschlange von Ereignisse, in der alle neu eingehenden Ereignisse hinzugefügt werden. Wenn ein [Timer](#)-Ereignis in der Warteschlange ist oder behandelt wird, wird die neue Timer-Ereignis nicht in der Warteschlange des MQL5-Programms platziert.

## EventKillTimer

Gibt dem Kundenterminal an, dass die Generierung der Ereignisse vom [Timer](#) für den Expert oder Indikator gestoppt werden muss.

```
void EventKillTimer();
```

### Rückgabewert

Keinen Rückgabewert.

### Bemerkung

Normalerweise muss diese Funktion aus der Funktion [OnDeinit\(\)](#) aufgerufen werden, falls die Funktion [EventSentTimer\(\)](#) in der Funktion [OnInit\(\)](#) aufgerufen wurde. Oder sie muss vom Destruktor der Klasse aufgerufen werden, wenn die Funktion EventSetTimer() im [Konstruktor](#) dieser Klasse aufgerufen wird.

Jeder Expert und jeder Indikator arbeitet nur mit seine Timer und erhaelt Ereignisse nur von ihm. Bei der Beendung des mql5-Programms wird Timer zwangslaeufig vernichtet, falls er erzeugt wurde aber nicht durch die Funktion EventKillTimer() ausgeschaltet wurde.

## EventChartCustom

Generiert ein Benutzerereignis für den angegebenen Chart.

```
bool EventChartCustom(
    long    chart_id,          // Identifikator des Charts-Ereignisrezipient
    ushort  custom_event_id,   // Identifikator des Ereignisses
    long    lparam,           // Parameter des Typs long
    double  dparam,           // Parameter des Typs double
    string  sparam            // Zeilenparameter des Ereignisses
);
```

### Parameter

*chart\_id*

[in] Identifikatoren des Charts. 0 bedeutet den laufenden Chart.

*custom\_event\_id*

[in] Identifikator des Benutzerereignisses. Dieser Identifikator wird automatisch zum Wert [CHARTEVENT\\_CUSTOM](#) hinzugefügt und zum ganzzahligen Typ reduziert.

*lparam*

[in] Ereignisparameter des Typs long, der der Funktion [OnChartEvent](#) übertragen wird.

*dparam*

[in] Ereignisparameter des Typs double, der der Funktion [OnChartEvent](#) übertragen wird.

*sparam*

[in] Parameter des Ereignisses des Typs string, der der Funktion [OnChartEvent](#) übertragen wird. Wenn die Zeile länger als 63 Symbole, wird die Zeile reduziert.

### Rückgabewert

Gibt true zurück, wenn ein benutzerdefiniertes Ereignis wurde erfolgreich in der Warteschlange von Ereignissen des Charts-Ereignisrezipient, platziert. Im Falle eines Fehlers, wird false zurückgegeben. Verwenden Sie [GetLastError\(\)](#), um einen Fehlercode zu bekommen.

### Bemerkung

Expert oder Indikator, angehängt zum angegebenen Chart, verarbeitet dieses Ereignis durch die Funktion [OnChartEvent](#)(int event\_id, long& lparam, double& dparam, string& sparam).

Für jede Art von Ereignis, Eingabeparameter der Funktion [OnChartEvent\(\)](#) haben bestimmte Werte, die nötig sind, um das Ereignis zu behandeln. Die folgende Tabelle listet die Ereignisse und Werte, die durch Parameter übergeben werden.

| Ereignis                       | Wert des Parameters id | Wert des Parameters lparam | Wert des Parameters dparam                    | Wert des Parameters sparam                      |
|--------------------------------|------------------------|----------------------------|-----------------------------------------------|-------------------------------------------------|
| Ereignis des Keyboard Klickens | CHARTEVENT_KEYDOWN     | Kode der betätigten Taste  | Die Anzahl der Tastenanschläge, die generiert | Der String-Wert eines Bit-Maske, die den Status |

|                                                                                                                                                     |                          |              |                                                       |                                                                           |
|-----------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------|--------------|-------------------------------------------------------|---------------------------------------------------------------------------|
|                                                                                                                                                     |                          |              | werden, während diese Taste in gedrücktem Zustand war | der Tastatur-Tasten beschreibt                                            |
| Ereignisse der Maus-Bewegung und Mausklicks (wenn die Eigenschaft <a href="#">CHART_EVENT_MOUSE_MOVE</a> =true ist für den Chart eingegeben)        | CHARTEVENT_MOUSE_MOVE    | X-Koordinate | Y-Koordinate                                          | Der String-Wert einer Bit-Maske, der den Status der Maustasten beschreibt |
| Ereignis der Erzeugung des graphischen Objektes (wenn die Eigenschaft <a href="#">CHART_EVENT_OBJECT_CREATE</a> =true ist für den Chart eingegeben) | CHARTEVENT_OBJECT_CREATE | —            | —                                                     | Name des erzeugten graphischen Objektes                                   |
| Ereignis der Veränderung der Eigenschaften des Objektes durch Dialog der Eigenschaften                                                              | CHARTEVENT_OBJECT_CHANGE | —            | —                                                     | Name des veränderten graphischen Objektes                                 |
| Ereignis der Löschung des graphischen Objektes (wenn die Eigenschaft <a href="#">CHART_EVENT_OBJECT_DELETE</a> =true ist für den Chart eingegeben)  | CHARTEVENT_OBJECT_DELETE | —            | —                                                     | Name des entfernten graphischen Objektes                                  |
| Ereignis des Mausklicks auf der graphischen Darstellung                                                                                             | CHARTEVENT_CLICK         | X-Koordinate | Y-Koordinate                                          | —                                                                         |
| Ereignis des Mausklicks auf dem graphischen Objekt                                                                                                  | CHARTEVENT_OBJECT_CLICK  | X-Koordinate | Y-Koordinate                                          | Name des graphischen Objektes, auf dem das Ereignis                       |

|                                                                                                 |                           |                                                                        |                                                                        |                                                                                        |
|-------------------------------------------------------------------------------------------------|---------------------------|------------------------------------------------------------------------|------------------------------------------------------------------------|----------------------------------------------------------------------------------------|
|                                                                                                 |                           |                                                                        |                                                                        | geschehen ist                                                                          |
| Ereignis der Verschiebung des graphischen Objektes per Mausklick                                | CHARTEVENT_OBJECT_DRAG    | —                                                                      | —                                                                      | Name des verschobenen graphischen Objektes                                             |
| Ereignis der Beendung der Textbearbeitung im Eingabefeld des graphischen Objektes "Eingabefeld" | CHARTEVENT_OBJECT_ENDEDIT | —                                                                      | —                                                                      | Name des graphischen Objektes "Eingabefeld", in dem Editieren des Textes beendet wurde |
| Ereignis der Chart-Veränderung                                                                  | CHARTEVENT_CHART_CHANGE   | —                                                                      | —                                                                      | —                                                                                      |
| Benutzerereignis mit Nummer N                                                                   | CHARTEVENT_CUSTOM+N       | Wert, vorgegeben durch die Funktion <a href="#">EventChartCustom()</a> | Wert, vorgegeben durch die Funktion <a href="#">EventChartCustom()</a> | Wert, vorgegeben durch die Funktion <a href="#">EventChartCustom()</a>                 |

Beispiel:

```

//+-----+
//|                                     ButtonClickExpert.mq5 |
//|                                     Copyright 2009, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

string buttonID="Button";
string labelID="Info";
int broadcastEventID=5000;
//+-----+
//| Expert initialization function |
//+-----+
int OnInit()
{
//--- Erzeugen wir einen Button für Übertragung der Benutzerereignisse
    ObjectCreate(0,buttonID,OBJ_BUTTON,0,100,100);
    ObjectSetInteger(0,buttonID,OBJPROP_COLOR,clrWhite);
    ObjectSetInteger(0,buttonID,OBJPROP_BGCOLOR,clrGray);
    ObjectSetInteger(0,buttonID,OBJPROP_XDISTANCE,100);
    ObjectSetInteger(0,buttonID,OBJPROP_YDISTANCE,100);
    ObjectSetInteger(0,buttonID,OBJPROP_XSIZE,200);
    ObjectSetInteger(0,buttonID,OBJPROP_YSIZE,50);
    ObjectSetString(0,buttonID,OBJPROP_FONT,"Arial");
    ObjectSetString(0,buttonID,OBJPROP_TEXT,"Button");
    ObjectSetInteger(0,buttonID,OBJPROP_FONTSIZE,10);
    ObjectSetInteger(0,buttonID,OBJPROP_SELECTABLE,0);

//--- Erzeugen wir eine Marke für Ausgabe der Information
    ObjectCreate(0,labelID,OBJ_LABEL,0,100,100);
    ObjectSetInteger(0,labelID,OBJPROP_COLOR,clrRed);
    ObjectSetInteger(0,labelID,OBJPROP_XDISTANCE,100);
    ObjectSetInteger(0,labelID,OBJPROP_YDISTANCE,50);
    ObjectSetString(0,labelID,OBJPROP_FONT,"Trebuchet MS");
    ObjectSetString(0,labelID,OBJPROP_TEXT,"Keine Information");
    ObjectSetInteger(0,labelID,OBJPROP_FONTSIZE,20);
    ObjectSetInteger(0,labelID,OBJPROP_SELECTABLE,0);

//---
    return(INIT_SUCCEEDED);
}
//+-----+
//| Expert deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
//---
    ObjectDelete(0,buttonID);
    ObjectDelete(0,labelID);
}
//+-----+
//| Expert tick function |
//+-----+
void OnTick()
{
//---

}
//+-----+

```

```

void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
{
    //--- Prüfen wir das Ereignis durch Drücken des Mausbuttons
    if(id==CHARTEVENT_OBJECT_CLICK)
    {
        string clickedChartObject=sparam;
        //--- wenn Sie drücken das Objekt mit dem Namen buttonID
        if(clickedChartObject==buttonID)
        {
            //--- Buttonstatus - gedrückt oder nicht
            bool selected=ObjectGetInteger(0,buttonID,OBJPROP_STATE);
            //--- geben wir in log Debugging-Nachricht aus
            Print("Knopf gedrückt = ",selected);
            int customEventID; // Nummer des Benutzerereignisses für Senden
            string message;    // Nachricht für Senden im Ereignis
            //--- wenn Knopf gedrückt ist
            if(selected)
            {
                message="Knopf gedrückt";
                customEventID=CHARTEVENT_CUSTOM+1;
            }
            else // Knopf nicht gedrückt
            {
                message="Knopf nicht gedrückt";
                customEventID=CHARTEVENT_CUSTOM+999;
            }
            //--- senden wir Benutzerereignis "unserem" Chart
            EventChartCustom(0,customEventID-CHARTEVENT_CUSTOM,0,0,message);
            //--- senden wir eine Nachricht an alle offenen Charts
            BroadcastEvent(ChatID(),0,"Broadcast Message");
            //--- Debugging-Nachricht
            Print("Ereignis mit ID ist gesendet = ",customEventID);
        }
        ChartRedraw(); // zeichnen wir alle Objekte auf dem Chart erneut
    }

    //--- prüfen wir, ob das Ereignis zu Benutzerereignissen gehört
    if(id>CHARTEVENT_CUSTOM)
    {
        if(id==broadcastEventID)
        {
            Print("Erhalten wir Broadcast-Nachricht vom Chart mit id = "+lparam);
        }
        else
        {
            //--- lesen wir eine Textnachricht im Ereignis
            string info=sparam;
            Print("Das Benutzerereignis mit ID wird verarbeitet = ",id);
            //--- geben wir eine Nachricht in der Marke aus
            ObjectSetString(0,labelID,OBJPROP_TEXT,sparam);
            ChartRedraw(); // zeichnen wir alle Objekte auf dem Chart zwangsläufig erneut
        }
    }
}

//+-----+
//| Broadcast-Nachricht allen offenen Charts senden |
//+-----+
void BroadcastEvent(long lparam,double dparam,string sparam)

```



```
{
    int eventID=broadcastEventID-CHARTEVENT_CUSTOM;
    long currChart=ChartFirst();
    int i=0;
    while(i<CHARTS_MAX)          // Sie haben bestimmt nicht mehr als CHARTS_MAX offene
    {
        EventChartCustom(currChart,eventID,lparam,dparam,sparam);
        currChart=ChartNext(currChart); // auf Grund des vorigen Charts erhalten wir ein
        if(currChart== -1) break;        // das Ende der Chartliste erreicht
        i++;                             // Nicht vergessen, Counter zu erhoeuen
    }
}
//+-----+
```

Sehen Sie auch

[Ereignisse des Kundenterminals](#), [Ereignisbearbeiter](#)

## Arbeit mit OpenCL

Programme auf [OpenCL](#) sind entwickelt, um Berechnungen auf Grafikkarten mit Unterstützung für OpenCL 1.1 oder höher auszuführen. Moderne Grafikkarten enthalten hunderte von kleinen, spezialisierten Prozessoren, die gleichzeitig einfache mathematische Operationen auf eingehenden Datenströme ausführen können. Die Sprache OpenCL übernimmt die Organisation der Parallel-Computing und ermöglicht eine hohe Geschwindigkeit für eine bestimmte Klasse von Aufgaben.

Funktionen zum Ausführen von Programmen auf OpenCL:

| Funktion                          | Aktion                                                                                                  |
|-----------------------------------|---------------------------------------------------------------------------------------------------------|
| <a href="#">CLHandleType</a>      | Gibt den Typ des OpenCL-Handles als ein Wert aus Aufzählung <code>ENUM_OPENCL_HANDLE_TYPE</code> zurück |
| <a href="#">CLGetInfoInteger</a>  | Gibt den Wert des ganzzahligen Eigenschafts für ein OpenCL-Objekt oder Gerät zurück                     |
| <a href="#">CLContextCreate</a>   | Erstellt einen Kontext für OpenCL                                                                       |
| <a href="#">CLContextFree</a>     | Löscht den Kontext von OpenCL                                                                           |
| <a href="#">CLGetDeviceInfo</a>   | Erhielt die Geräteeigenschaft aus dem OpenCL-Treiber                                                    |
| <a href="#">CLProgramCreate</a>   | Erstellt ein OpenCL-Programm aus dem Quellcode                                                          |
| <a href="#">CLProgramFree</a>     | Löscht das OpenCL-Programm                                                                              |
| <a href="#">CLKernelCreate</a>    | Erstellt eine Funktion des Starts von OpenCL                                                            |
| <a href="#">CLKernelFree</a>      | Löscht die Funktion des Starts von OpenCL                                                               |
| <a href="#">CLSetKernelArg</a>    | Setzt den Parameter für die OpenCL-Funktion                                                             |
| <a href="#">CLSetKernelArgMem</a> | Setzt einen OpenCL-Puffer als ein Parameter der OpenCL-Funktion                                         |
| <a href="#">CLBufferCreate</a>    | Erstellt einen OpenCL-Puffer                                                                            |
| <a href="#">CLBufferFree</a>      | Löscht einen OpenCL-Puffer                                                                              |
| <a href="#">CLBufferWrite</a>     | Schreibt ein Array in einen OpenCL-Puffer                                                               |
| <a href="#">CLBufferRead</a>      | Liest einen OpenCL-Puffer in ein Array                                                                  |
| <a href="#">CLExecute</a>         | Führt OpenCL-Programm                                                                                   |

## CLHandleType

Gibt den Typ des OpenCL-Handles als ein Wert aus Aufzählung `ENUM_OPENCL_HANDLE_TYPE` zurück.

```
ENUM_OPENCL_HANDLE_TYPE CLHandleType(  
    int handle    // Handle des OpenCL-Objekts  
);
```

### Parameter

*handle*

[in] Handle auf ein OpenCL-Objekt: Kontext, Kernel, Puffer oder OpenCL-Programm.

### Rückgabewert

Typ des OpenCL-Handles als ein Wert aus Aufzählung [ENUM\\_OPENCL\\_HANDLE\\_TYPE](#).

### ENUM\_OPENCL\_HANDLE\_TYPE

| Identifizier   | Beschreibung               |
|----------------|----------------------------|
| OPENCL_INVALID | Ungültiges Handle          |
| OPENCL_CONTEXT | Handle von OpenCL-Kontext  |
| OPENCL_PROGRAM | Handle von OpenCL-Programm |
| OPENCL_KERNEL  | Handle von OpenCL-Kernel   |
| OPENCL_BUFFER  | Handle von OpenCL-Puffer   |

## CLGetInfoInteger

Gibt den Wert der ganzzahligen Eigenschaft für ein OpenCL-Objekt oder Gerät zurück.

```
long CLGetInfoInteger(  
    int handle, // Handle des OpenCL-Objekts oder Nummer des  
    ENUM_OPENCL_PROPERTY_INTEGER prop // Die angeforderte Eigenschaft  
);
```

### Parameter

*handle*

[in] Handle auf ein OpenCL-Objekt oder Nummer des OpenCL-Geräts. Nummerierung der OpenCL-Geräte startet mit Null.

*prop*

[in] Typ der angeforderten Eigenschaft von der Listing [ENUM\\_OPENCL\\_PROPERTY\\_INTEGER](#), deren Wert Sie möchten erhalten.

### Rückgabewert

Der Wert der angegebenen Eigenschaft beim Erfolg oder -1 bei einem Fehler. Um Informationen über den Fehler zu erhalten, rufen Sie die Funktion [GetLastError\(\)](#).

### ENUM\_OPENCL\_PROPERTY\_INTEGER

| Identifikator                 | Beschreibung                                                                                                                                                                                   | Typ                                 |
|-------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------|
| CL_DEVICE_COUNT               | Die Anzahl der Geräte mit Unterstützung für OpenCL. Diese Eigenschaft erfordert nicht Eingabe des ersten Parameter, ist es möglich, einen Null-Wert für den <i>handle</i> Parameter anzugeben. | int                                 |
| CL_DEVICE_TYPE                | Gerätetyp                                                                                                                                                                                      | <a href="#">ENUM_CL_DEVICE_TYPE</a> |
| CL_DEVICE_VENDOR_ID           | Eindeutiger Identifikator des Verkäufers                                                                                                                                                       | uint                                |
| CL_DEVICE_MAX_COMPUTE_UNITS   | Anzahl der parallel berechneten Aufgaben im OpenCL-Gerät. Eine Arbeitsgruppe erfüllt eine Rechenaufgabe. Der Mindestwert ist 1.                                                                | uint                                |
| CL_DEVICE_MAX_CLOCK_FREQUENCY | Maximale Frequenz des Gerätes in MHz.                                                                                                                                                          | uint                                |
| CL_DEVICE_GLOBAL_MEM_SIZE     | Größe des globalen Speichers des Gerätes in Bytes                                                                                                                                              | ulong                               |
| CL_DEVICE_LOCAL_MEM_SIZE      | Größe des lokalen Speichers                                                                                                                                                                    | uint                                |

|  |                                             |  |
|--|---------------------------------------------|--|
|  | der verarbeiteten Daten<br>(Szene) in Bytes |  |
|--|---------------------------------------------|--|

Aufzählung `ENUM_CL_DEVICE_TYPE` enthält Typen der Geräte mit Unterstützung für OpenCL. Gerätetyp kann durch seine Nummer oder Handle des Objekts OpenCL durch den Aufruf von `CLGetInfoInteger(handle_or_deviceN, CL_DEVICE_TYPE)` erhalten werden.

ENUM\_CL\_DEVICE\_TYPE

| Identifikator         | Beschreibung                                                                                                                            |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| CL_DEVICE_ACCELERATOR | Spezialisierter OpenCL-Beschleuniger (zum Beispiel IBM CELL Blade).                                                                     |
| CL_DEVICE_CPU         | Das OpenCL-Gerät ist ein Host-Prozessor. Der Host-Prozessor läuft die OpenCL-Implementierungen und ist ein Single- oder Multi-Core-CPU. |
| CL_DEVICE_GPU         | Das OpenCL-Gerät ist eine GPU.                                                                                                          |
| CL_DEVICE_DEFAULT     | Das Standardgerät OpenCL. Das Standardgerät kann nicht ein CL_DEVICE_TYPE_CUSTOM-Gerät sein.                                            |
| CL_DEVICE_CUSTOM      | Spezielle Beschleunigern, die Programme in OpenCL C nicht unterstützen.                                                                 |

Beispiel:

```
void OnStart()
{
    int cl_ctx;
    ///--- Initialisierung von OpenCL Kontext
    if ((cl_ctx=CLContextCreate(CL_USE_GPU_ONLY))==INVALID_HANDLE)
    {
        Print("OpenCL not found");
        return;
    }
    ///--- Allgemeine Informationen über das OpenCL Gerät anzeigen
    Print("OpenCL type: ",EnumToString((ENUM_CL_DEVICE_TYPE)CLGetInfoInteger(cl_ctx,CL_
    Print("OpenCL vendor ID: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_VENDOR_ID));
    Print("OpenCL units: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_COMPUTE_UNITS));
    Print("OpenCL freq: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_CLOCK_FREQUENCY)," MHz");
    Print("OpenCL global mem: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_GLOBAL_MEM_SIZE)," by
    Print("OpenCL local mem: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_LOCAL_MEM_SIZE)," byte
    ///---
}
```

## CLGetInfoString

Gibt den string Wert des Eigenschafts für ein OpenCL-Objekt oder Gerät zurück.

```
bool CLGetInfoString(  
    int handle, // Handle des OpenCL-Objekts oder Nummer des  
    ENUM_OPENCL_PROPERTY_STRING prop, // Die angeforderte Eigenschaft  
    string& value // String  
);
```

### Optionen

*handle*

[in] Handle auf ein OpenCL-Objekt oder Nummer des OpenCL-Geräts. Nummerierung der OpenCL-Geräte startet mit Null.

*prop*

[in] Typ der angeforderten Eigenschaft von der Aufzählung [ENUM\\_OPENCL\\_PROPERTY\\_STRING](#), deren Wert Sie erhalten möchten.

*value*

[out] String den Wert der Eigenschaft zu erhalten.

### Rückgabewert

true beim Erfolg oder false beim Fehler. Um Informationen über den Fehler zu erhalten, rufen Sie die Funktion [GetLastError\(\)](#).

### ENUM\_OPENCL\_PROPERTY\_STRING

| Identifizier           | Beschreibung                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CL_PLATFORM_PROFILE    | CL_PLATFORM_PROFILE - Profil von OpenCL. Der Profilname kann ein der folgenden Werte sein: <ul style="list-style-type: none"><li>FULL_PROFILE - Implementierung unterstützt OpenCL (Funktionalität ist als Teil der Kernspezifikation definiert und erfordert keine zusätzlichen Erweiterungen für OpenCL Unterstützung);</li><li>EMBEDDED_PROFILE - Implementierung unterstützt OpenCL in Form einer Ergänzung. Ergänzttes Profil wird als Teilmenge für jede Version von OpenCL definiert.</li></ul> |
| CL_PLATFORM_VERSION    | OpenCL-Version                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| CL_PLATFORM_VENDOR     | Name des Geräteherstellers                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| CL_PLATFORM_EXTENSIONS | Liste der von der Plattform unterstützten Erweiterungen. Erweiterungsennamen müssen von allen Geräten, die mit dieser Plattform verbunden sind, unterstützt werden                                                                                                                                                                                                                                                                                                                                     |

|                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CL_DEVICE_NAME             | Gerätename                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| CL_DEVICE_VENDOR           | Name des Herstellers                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| CL_DRIVER_VERSION          | Version von OpenCL-Treiber in Format major_number.minor_number                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| CL_DEVICE_PROFILE          | <p>Profil des OpenCL-Gerätes. Der Profilname kann ein der folgenden Werte sein:</p> <ul style="list-style-type: none"> <li>• FULL_PROFILE - Implementierung unterstützt OpenCL (Funktionalität ist als Teil der Kernspezifikation definiert und erfordert keine zusätzlichen Erweiterungen für OpenCL Unterstützung);</li> <li>• EMBEDDED_PROFILE - Implementierung unterstützt OpenCL in Form einer Ergänzung. Ergänzt Profil wird als Teilmenge für jede Version von OpenCL definiert.</li> </ul> |
| CL_DEVICE_VERSION          | Version von OpenCL in Format "OpenCL<space><major_version.minor_version><space><vendor-specific information>"                                                                                                                                                                                                                                                                                                                                                                                       |
| CL_DEVICE_EXTENSIONS       | <p>Liste der vom Gerät unterstützten Erweiterungen. Die Liste kann vom Hersteller unterstützten Erweiterungen erhalten und kann einen oder mehrere zugelassene Namen haben:</p> <pre> cl_khr_int64_base_atomics cl_khr_int64_extended_atomics cl_khr_fp16 cl_khr_gl_sharing cl_khr_gl_event cl_khr_d3d10_sharing cl_khr_dx9_media_sharing cl_khr_d3d11_sharing </pre>                                                                                                                               |
| CL_DEVICE_BUILT_IN_KERNELS | Liste der vom Gerät unterstützten Kernel, getrennten mit ";".                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| CL_DEVICE_OPENCL_C_VERSION | <p>Maximale Version, die vom Compiler für dieses Gerät unterstützt ist. Format der Version:</p> <p>"OpenCL&lt;space&gt;C&lt;space&gt;&lt;major_version.minor_version&gt;&lt;space&gt;&lt;vendor-specific information&gt; "</p>                                                                                                                                                                                                                                                                      |

Beispiel:

```

void OnStart()
{
    int cl_ctx;
    string str;
    //--- Initialisierung von OpenCL Kontext
    if((cl_ctx=CLContextCreate(CL_USE_GPU_ONLY))==INVALID_HANDLE)
    {
        Print("OpenCL not found");
        return;
    }
    //--- Information über die Plattform anzeigen
    if(CLGetInfoString(cl_ctx,CL_PLATFORM_NAME,str))
        Print("OpenCL platform name: ",str);
    if(CLGetInfoString(cl_ctx,CL_PLATFORM_VENDOR,str))
        Print("OpenCL platform vendor: ",str);
    if(CLGetInfoString(cl_ctx,CL_PLATFORM_VERSION,str))
        Print("OpenCL platform ver: ",str);
    if(CLGetInfoString(cl_ctx,CL_PLATFORM_PROFILE,str))
        Print("OpenCL platform profile: ",str);
    if(CLGetInfoString(cl_ctx,CL_PLATFORM_EXTENSIONS,str))
        Print("OpenCL platform ext: ",str);
    //--- Information über das Gerät anzeigen
    if(CLGetInfoString(cl_ctx,CL_DEVICE_NAME,str))
        Print("OpenCL device name: ",str);
    if(CLGetInfoString(cl_ctx,CL_DEVICE_PROFILE,str))
        Print("OpenCL device profile: ",str);
    if(CLGetInfoString(cl_ctx,CL_DEVICE_BUILT_IN_KERNELS,str))
        Print("OpenCL device kernels: ",str);
    if(CLGetInfoString(cl_ctx,CL_DEVICE_EXTENSIONS,str))
        Print("OpenCL device ext: ",str);
    if(CLGetInfoString(cl_ctx,CL_DEVICE_VENDOR,str))
        Print("OpenCL device vendor: ",str);
    if(CLGetInfoString(cl_ctx,CL_DEVICE_VERSION,str))
        Print("OpenCL device ver: ",str);
    if(CLGetInfoString(cl_ctx,CL_DEVICE_OPENCL_C_VERSION,str))
        Print("OpenCL open c ver: ",str);
    //--- Gemeinsame Information über das OpenCL-Gerät anzeigen
    Print("OpenCL type: ",EnumToString((ENUM_CL_DEVICE_TYPE)CLGetInfoInteger(cl_ctx,CL_
    Print("OpenCL vendor ID: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_VENDOR_ID));
    Print("OpenCL units: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_COMPUTE_UNITS));
    Print("OpenCL freq: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_CLOCK_FREQUENCY));
    Print("OpenCL global mem: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_GLOBAL_MEM_SIZE));
    Print("OpenCL local mem: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_LOCAL_MEM_SIZE));
    //---
}

```



## CLContextCreate

Erstellt einen OpenCL-Kontext und gibt sein Handle zurück.

```
int CLContextCreate(  
    int device    // Seriennummer von OpenCL-Gerät oder Makro  
);
```

### Parameter

*device*

[in] Seriennummer von OpenCL-Gerät. Anstatt einer bestimmten Nummer können Sie einen der Werte angeben:

- CL\_USE\_ANY - ein beliebiges Gerät mit Unterstützung für OpenCL genutzt werden kann;
- CL\_USE\_CPU\_ONLY - nur eine Emulation von OpenCL auf CPU ist erlaubt;
- CL\_USE\_GPU\_ONLY - emulation von OpenCL ist verboten und es darf nur spezialisierte Geräte mit Unterstützung für OpenCL (Grafikkarte) zu verwenden.

### Rückgabewert

Handle auf den OpenCL-Kontext für den Erfolg, oder -1 bei einem Fehler. Um Informationen über den Fehler zu erhalten, rufen Sie die Funktion [GetLastError\(\)](#).

## CLContextFree

Löscht den Kontext von OpenCL.

```
void CLContextFree(  
    int context    // Handle auf den OpenCL-Kontext  
);
```

### Parameter

*context*

[in] Handle auf den OpenCL-Kontext.

### Rückgabewert

Kein. Im Falle eines internen Fehlers, ändert sich der Wert von [\\_LastError](#). Um Informationen über den Fehler zu erhalten, rufen Sie die Funktion [GetLastError\(\)](#).

## CLGetDeviceInfo

Erhielt die Geräteeigenschaft aus dem OpenCL-Treiber.

```
bool CLGetDeviceInfo(  
    int      handle,          // Handle von OpenCL-Gerät  
    int      property_id,     // Identifikator der gefragten Eigenschaft  
    uchar&   data[],         // Array für Daten  
    uint&    size             // Offset im Array in Elemente, standardmäßig 0  
);
```

### Optionen

*handle*

[in] Nummer von OpenCL-Gerät oder OpenCL-Handle, der durch die Funktion [CLContextCreate\(\)](#) erstellt wurde.

*property\_id*

[in] Identifikator der Eigenschaft, die Sie über das OpenCL-Gerät erhalten möchten. Kann einer der vordefinierten Werten aus der [unteren Tabelle](#) sein.

*data[]*

[out] Array, um die Daten über das gefragte Gerät zu erhalten.

*size*

[out] Die Größe der erhaltenen Daten im Array *data[]*.

### Rückgabewert

true beim Erfolg oder false beim Fehler. Um Informationen über den Fehler zu erhalten, rufen Sie die Funktion [GetLastError\(\)](#).

### Hinweis

Für eindimensionale Arrays, wird die Nummer des Elements, mit dem Lesung von Daten für Schreibung in den OpenCL-Puffer beginnt, unter Berücksichtigung der Flagge [AS\\_SERIES](#) ausgerechnet.

### Liste der gültigen Eigenschaftenbezeichner von OpenCL-Gerät

Die genaue Beschreibung der Eigenschaft und ihre Werte finden Sie auf die [offizielle OpenCL Website](#).

| Identifizier                       | Wert   |
|------------------------------------|--------|
| CL_DEVICE_TYPE                     | 0x1000 |
| CL_DEVICE_VENDOR_ID                | 0x1001 |
| CL_DEVICE_MAX_COMPUTE_UNITS        | 0x1002 |
| CL_DEVICE_MAX_WORK_ITEM_DIMENSIONS | 0x1003 |
| CL_DEVICE_MAX_WORK_GROUP_SIZE      | 0x1004 |
| CL_DEVICE_MAX_WORK_ITEM_SIZES      | 0x1005 |

|                                            |        |
|--------------------------------------------|--------|
| CL_DEVICE_PREFERRED_VECTOR_WIDTH_CHARACTER | 0x1006 |
| CL_DEVICE_PREFERRED_VECTOR_WIDTH_SHORT     | 0x1007 |
| CL_DEVICE_PREFERRED_VECTOR_WIDTH_INT       | 0x1008 |
| CL_DEVICE_PREFERRED_VECTOR_WIDTH_LONG      | 0x1009 |
| CL_DEVICE_PREFERRED_VECTOR_WIDTH_FLOAT     | 0x100A |
| CL_DEVICE_PREFERRED_VECTOR_WIDTH_DOUBLE    | 0x100B |
| CL_DEVICE_MAX_CLOCK_FREQUENCY              | 0x100C |
| CL_DEVICE_ADDRESS_BITS                     | 0x100D |
| CL_DEVICE_MAX_READ_IMAGE_ARGS              | 0x100E |
| CL_DEVICE_MAX_WRITE_IMAGE_ARGS             | 0x100F |
| CL_DEVICE_MAX_MEM_ALLOC_SIZE               | 0x1010 |
| CL_DEVICE_IMAGE2D_MAX_WIDTH                | 0x1011 |
| CL_DEVICE_IMAGE2D_MAX_HEIGHT               | 0x1012 |
| CL_DEVICE_IMAGE3D_MAX_WIDTH                | 0x1013 |
| CL_DEVICE_IMAGE3D_MAX_HEIGHT               | 0x1014 |
| CL_DEVICE_IMAGE3D_MAX_DEPTH                | 0x1015 |
| CL_DEVICE_IMAGE_SUPPORT                    | 0x1016 |
| CL_DEVICE_MAX_PARAMETER_SIZE               | 0x1017 |
| CL_DEVICE_MAX_SAMPLERS                     | 0x1018 |
| CL_DEVICE_MEM_BASE_ADDR_ALIGN              | 0x1019 |
| CL_DEVICE_MIN_DATA_TYPE_ALIGN_SIZE         | 0x101A |
| CL_DEVICE_SINGLE_FP_CONFIG                 | 0x101B |
| CL_DEVICE_GLOBAL_MEM_CACHE_TYPE            | 0x101C |
| CL_DEVICE_GLOBAL_MEM_CACHELINE_SIZE        | 0x101D |
| CL_DEVICE_GLOBAL_MEM_CACHE_SIZE            | 0x101E |
| CL_DEVICE_GLOBAL_MEM_SIZE                  | 0x101F |
| CL_DEVICE_MAX_CONSTANT_BUFFER_SIZE         | 0x1020 |
| CL_DEVICE_MAX_CONSTANT_ARGS                | 0x1021 |
| CL_DEVICE_LOCAL_MEM_TYPE                   | 0x1022 |

|                                       |        |
|---------------------------------------|--------|
| CL_DEVICE_LOCAL_MEM_SIZE              | 0x1023 |
| CL_DEVICE_ERROR_CORRECTION_SUPPORT    | 0x1024 |
| CL_DEVICE_PROFILING_TIMER_RESOLUTION  | 0x1025 |
| CL_DEVICE_ENDIAN_LITTLE               | 0x1026 |
| CL_DEVICE_AVAILABLE                   | 0x1027 |
| CL_DEVICE_COMPILER_AVAILABLE          | 0x1028 |
| CL_DEVICE_EXECUTION_CAPABILITIES      | 0x1029 |
| CL_DEVICE_QUEUE_PROPERTIES            | 0x102A |
| CL_DEVICE_NAME                        | 0x102B |
| CL_DEVICE_VENDOR                      | 0x102C |
| CL_DRIVER_VERSION                     | 0x102D |
| CL_DEVICE_PROFILE                     | 0x102E |
| CL_DEVICE_VERSION                     | 0x102F |
| CL_DEVICE_EXTENSIONS                  | 0x1030 |
| CL_DEVICE_PLATFORM                    | 0x1031 |
| CL_DEVICE_DOUBLE_FP_CONFIG            | 0x1032 |
| CL_DEVICE_PREFERRED_VECTOR_WIDTH_HALF | 0x1034 |
| CL_DEVICE_HOST_UNIFIED_MEMORY         | 0x1035 |
| CL_DEVICE_NATIVE_VECTOR_WIDTH_CHAR    | 0x1036 |
| CL_DEVICE_NATIVE_VECTOR_WIDTH_SHORT   | 0x1037 |
| CL_DEVICE_NATIVE_VECTOR_WIDTH_INT     | 0x1038 |
| CL_DEVICE_NATIVE_VECTOR_WIDTH_LONG    | 0x1039 |
| CL_DEVICE_NATIVE_VECTOR_WIDTH_FLOAT   | 0x103A |
| CL_DEVICE_NATIVE_VECTOR_WIDTH_DOUBLE  | 0x103B |
| CL_DEVICE_NATIVE_VECTOR_WIDTH_HALF    | 0x103C |
| CL_DEVICE_OPENCL_C_VERSION            | 0x103D |
| CL_DEVICE_LINKER_AVAILABLE            | 0x103E |
| CL_DEVICE_BUILT_IN_KERNELS            | 0x103F |
| CL_DEVICE_IMAGE_MAX_BUFFER_SIZE       | 0x1040 |
| CL_DEVICE_IMAGE_MAX_ARRAY_SIZE        | 0x1041 |
| CL_DEVICE_PARENT_DEVICE               | 0x1042 |
| CL_DEVICE_PARTITION_MAX_SUB_DEVICES   | 0x1043 |

|                                            |        |
|--------------------------------------------|--------|
| CL_DEVICE_PARTITION_PROPERTIES             | 0x1044 |
| CL_DEVICE_PARTITION_AFFINITY_DOMAIN        | 0x1045 |
| CL_DEVICE_PARTITION_TYPE                   | 0x1046 |
| CL_DEVICE_REFERENCE_COUNT                  | 0x1047 |
| CL_DEVICE_PREFERRED_INTEROP_USER_SYNC      | 0x1048 |
| CL_DEVICE_PRINTF_BUFFER_SIZE               | 0x1049 |
| CL_DEVICE_IMAGE_PITCH_ALIGNMENT            | 0x104A |
| CL_DEVICE_IMAGE_BASE_ADDRESS_ALIGNMEN<br>T | 0x104B |

**Beispiel:**

```

void OnStart()
{
//---
    int dCount= CLGetInfoInteger(0,CL_DEVICE_COUNT);
    for(int i = 0; i<dCount; i++)
    {
        int clCtx=CLContextCreate(i);
        if(clCtx == -1)
            Print("ERROR in CLContextCreate");
        string device;
        CLGetInfoString(clCtx,CL_DEVICE_NAME,device);
        Print(i," : ",device);
        uchar data[1024];
        uint size;
        CLGetDeviceInfo(clCtx,CL_DEVICE_VENDOR,data,size);
        Print("size = ",size);
        string str=CharArrayToString(data);
        Print(str);
    }
};
//--- Beispiel der Ausgabe in Protokoll Experten
// 2013.07.24 10:50:48      openc1 (EURUSD,H1)      2: Advanced Micro Devices, Inc.
// 2013.07.24 10:50:48      openc1 (EURUSD,H1)      size = 32
// 2013.07.24 10:50:48      openc1 (EURUSD,H1)      Tahiti
// 2013.07.24 10:50:48      openc1 (EURUSD,H1)      Intel(R) Corporation
// 2013.07.24 10:50:48      openc1 (EURUSD,H1)      size = 21
// 2013.07.24 10:50:48      openc1 (EURUSD,H1)      1:          Intel(R) Core(TM) i7-377
// 2013.07.24 10:50:48      openc1 (EURUSD,H1)      NVIDIA Corporation
// 2013.07.24 10:50:48      openc1 (EURUSD,H1)      size = 19
// 2013.07.24 10:50:48      openc1 (EURUSD,H1)      0: GeForce GTX 580

```

## CLProgramCreate

Erstellt ein OpenCL-Programm aus dem Quellcode.

```
int CLProgramCreate(  
    int          context,    // Handle auf den OpenCL-Kontext  
    const string source      // Quellcode  
);
```

### Parameter

*context*

[in] Handle auf den OpenCL-Kontext.

*source*

[in] String mit dem Quellcode des OpenCL-Programms.

### Rückgabewert

Handle auf das OpenCL-Objekt für beim Erfolg. Im Falle eines Fehlers wird -1 zurückgegeben. Um Informationen über den Fehler zu erhalten, rufen Sie die Funktion [GetLastError\(\)](#).

### Bemerkung

Im Moment werden die folgenden Fehler-Codes verwendet:

- ERR\_OPENCL\_INVALID\_HANDLE - ungültiges Handle auf context OpenCL.
- ERR\_INVALID\_PARAMETER - ungültiger String-Parameter.
- ERR\_NOT\_ENOUGH\_MEMORY - nicht genügend Speicher um Operation abzuschließen.
- ERR\_OPENCL\_PROGRAM\_CREATE - interner Fehler von OpenCL oder Kompilierungsfehler.

## CLProgramFree

Löscht das OpenCL-Programm.

```
void CLProgramFree(  
    int program    // Handle auf das OpenCL-Objekt  
);
```

### Parameter

*program*

[in] Handle des OpenCL-Objekts.

### Rückgabewert

Kein. Im Falle eines internen Fehlers, ändert sich der Wert von [\\_LastError](#). Um Informationen über den Fehler zu erhalten, rufen Sie die Funktion [GetLastError\(\)](#).



## CLKernelCreate

Erstellt eine Einsprungstelle in das OpenCL-Programm und gibt sein Handle zurück.

```
int  CLKernelCreate(  
    int          program,          // Handle auf das OpenCL-Objekt  
    const string kernel_name      // Name des Kernels  
);
```

### Parameter

*program*

[in] Handle auf das Objekt von OpenCL-Programm.

*kernel\_name*

[in] Der Name der Kernel-Funktion, d.h. der Name der Einsprungstelle in das entsprechende OpenCL-Programm,, in dem Ausführung beginnt.

### Rückgabewert

Handle auf das OpenCL-Objekt für beim Erfolg. Im Falle eines Fehlers wird -1 zurückgegeben. Um Informationen über den Fehler zu erhalten, rufen Sie die Funktion [GetLastError\(\)](#).

### Bemerkung

Im Moment werden die folgenden Fehler-Codes verwendet:

- ERR\_OPENCL\_INVALID\_HANDLE - ungültiges Handle auf *program* OpenCL.
- ERR\_INVALID\_PARAMETER - ungültiger String-Parameter.
- ERR\_OPENCL\_TOO\_LONG\_KERNEL\_NAME - der Name des Kernels, enthält mehr als 127 Zeichen.
- ERR\_OPENCL\_KERNEL\_CREATE - Interner Fehler beim Erstellen von OpenCL-Objekt.

## CLKernelFree

Löscht die Funktion des Starts von OpenCL.

```
void CLKernelFree(  
    int kernel    // Handle auf den Kernel von OpenCL-Programm  
);
```

### Parameter

*kernel\_name*

[in] Handle des Objekts von Kernel.

### Rückgabewert

Kein. Im Falle eines internen Fehlers, ändert sich der Wert von [\\_LastError](#). Um Informationen über den Fehler zu erhalten, rufen Sie die Funktion [GetLastError\(\)](#).

## CLSetKernelArg

Setzt den Parameter für die OpenCL-Funktion.

```
bool CLSetKernelArg(  
    int    kernel,          // Handle auf den Kernel von OpenCL-Programm  
    uint   arg_index,      // Nummer des OpenCL-Funktionsarguments  
    void   arg_value       // Quellcode  
);
```

### Parameter

*kernel*

[in] Handle auf den Kernel von OpenCL-Programm.

*arg\_index*

[in] Nummer des Funktionsarguments. Nummerierung beginnt mit Null.

*arg\_value*

[in] Wert des Funktionsarguments.

### Rückgabewert

Gibt bei Erfolg true zurück, ansonsten false. Um Informationen über den Fehler zu erhalten, rufen Sie die Funktion [GetLastError\(\)](#).

### Bemerkung

Im Moment werden die folgenden Fehler-Codes verwendet:

- ERR\_INVALID\_PARAMETER,
- ERR\_OPENCL\_INVALID\_HANDLE - ungültiges Handle auf den OpenCL-Kernel
- ERR\_OPENCL\_SET\_KERNEL\_PARAMETER - interner Fehler von OpenCL.

## CLSetKernelArgMem

Setzt einen OpenCL-Puffer als ein Parameter der OpenCL-Funktion.

```
bool CLSetKernelArgMem(  
    int    kernel,           // Handle auf den Kernel von OpenCL-Programm  
    uint   arg_index,       // Nummer des OpenCL-Funktionsarguments  
    int    cl_mem_handle    // Handle des OpenCL-Puffers  
);
```

### Parameter

*kernel*

[in] Handle auf den Kernel von OpenCL-Programm.

*arg\_index*

[in] Nummer des Funktionsarguments. Nummerierung beginnt mit Null.

*cl\_mem\_handle*

[in] Handle auf den OpenCL-Puffer.

### Rückgabewert

Gibt bei Erfolg true zurück, ansonsten false. Um Informationen über den Fehler zu erhalten, rufen Sie die Funktion [GetLastError\(\)](#).

## CLBufferCreate

Erstellt einen OpenCL-Puffer und gibt sein Handle zurück.

```
int CLBufferCreate(  
    int    context,    // Handle auf den OpenCL-Kontext  
    uint   size,       // Größe des Puffers  
    uint   flags        // Kombination von Flags, die Eigenschaften des Puffers angeben  
);
```

### Parameter

*context*

[in] Handle auf den OpenCL-Kontext.

*size*

[in] Größe des Puffers in Bytes.

*flags*

[in] Eigenschaften des Puffers, sie durch eine Kombination von Flags definiert werden:  
CL\_MEM\_READ\_WRITE, CL\_MEM\_WRITE\_ONLY, CL\_MEM\_READ\_ONLY,  
CL\_MEM\_ALLOC\_HOST\_PTR.

### Rückgabewert

Handle auf den OpenCL-Puffer für den Erfolg. Im Falle eines Fehlers wird -1 zurückgegeben. Um Informationen über den Fehler zu erhalten, rufen Sie die Funktion [GetLastError\(\)](#).

### Bemerkung

Im Moment werden die folgenden Fehler-Codes verwendet:

- ERR\_OPENCL\_INVALID\_HANDLE - ungültiges Handle auf den OpenCL-Kontext.
- ERR\_NOT\_ENOUGH\_MEMORY - nicht genügend Speicher.
- ERR\_OPENCL\_BUFFER\_CREATE - interner Fehler beim Erstellen von Puffer.

## CLBufferFree

Löscht einen OpenCL-Puffer.

```
void CLBufferFree(  
    int    buffer    // Handle auf den OpenCL-Puffer  
);
```

### Parameter

*buffer*

[in] Handle auf den OpenCL-Puffer.

### Rückgabewert

Kein. Im Falle eines internen Fehlers, ändert sich der Wert von [\\_LastError](#). Um Informationen über den Fehler zu erhalten, rufen Sie die Funktion [GetLastError\(\)](#).

## CLBufferWrite

Schreibt ein Array in einen OpenCL-Puffer und gibt die Anzahl der geschriebenen Elemente zurück.

```
uint CLBufferWrite(  
    int          buffer,                // Handle auf den OpenCL-Puffer  
    const void&  data[],                // Array von Werten  
    uint         buffer_offset=0,       // Offset im OpenCL-Puffer in Bytes, standardmäßig 0  
    uint         data_offset=0,         // Offset im Array in Elemente, standardmäßig 0  
    uint         data_count=WHOLE_ARRAY // Anzahl der Werte im Array zu schreiben,  
);
```

### Parameter

*buffer*

[in] Handle auf den OpenCL-Puffer.

*data[]*

[in] Das Array von Werten, die Sie in den OpenCL-Puffer schreiben möchten. Wird als Referenz übergeben.

*buffer\_offset*

[in] Offset im OpenCL-Puffer, mit dem Schreibung beginnt, in Bytes. Standardmäßig wird vom Anfang des Puffers geschrieben.

*data\_offset*

[in] Der Index des ersten Elements des Arrays aus dem die Werte für Schreiben in den OpenCL-Puffer beginnen. Standardmäßig werden die Werte vom Anfang des Arrays genommen.

*data\_count*

[in] Die Anzahl der Werte zu schreiben. Standardmäßig werden alle Werte des Arrays genommen.

### Rückgabewert

Anzahl der geschriebenen Elemente. Im Falle eines Fehlers wird 0 zurückgegeben. Um Informationen über den Fehler zu erhalten, rufen Sie die Funktion [GetLastError\(\)](#).

### Bemerkung

Für eindimensionale Arrays, wird die Nummer des Elements, mit dem Lesung von Daten für Schreibung in den OpenCL-Puffer beginnt, unter Berücksichtigung der Flagge [AS\\_SERIES](#) ausgerechnet.

Ein Array von zwei oder mehr Dimensionen wird als eindimensional dargestellt. In diesem Fall ist *data\_Offset* die Anzahl der Elemente, die in der Darstellung angegeben werden soll, nicht die Anzahl der Elemente in der ersten Dimension.

## CLBufferRead

Liest den OpenCL-Puffer in ein Array und gibt die Anzahl der gelesenen Elemente zurück.

```
uint CLBufferRead(  
    int          buffer,           // Handle auf den OpenCL-Puffer  
    const void&  data[],          // Array von Werten  
    uint         buffer_offset=0,  // Offset im OpenCL-Puffer in Bytes, standardmäßig 0  
    uint         data_offset=0,    // Offset im Array in Elemente, standardmäßig 0  
    uint         data_count=WHOLE_ARRAY // Anzahl der Werte aus dem Puffer zu lesen  
);
```

### Parameter

*buffer*

[in] Handle auf den OpenCL-Puffer.

*data[]*

[in] Array, um die Werte aus dem OpenCL-Puffer zu empfangen. Wird als Referenz übergeben.

*buffer\_offset*

[in] Offset im OpenCL-Puffer, mit dem Lesung beginnt, in Bytes. Standardmäßig startet die Lesung am Anfang des Puffers.

*data\_offset*

[in] Der Index des ersten Elements des Arrays, um Werte des OpenCL-Puffers zu schreiben. Standardmäßig startet die Schreibung von gelesenen Werten in ein Array mit dem Null-Index.

*data\_count*

[in] Die Anzahl der Werte zu lesen. Standardmäßig wird der gesamte OpenCL-Puffer gelesen.

### Rückgabewert

Anzahl der gelesenen Elemente. Im Falle eines Fehlers wird 0 zurückgegeben. Um Informationen über den Fehler zu erhalten, rufen Sie die Funktion [GetLastError\(\)](#).

### Bemerkung

Für eindimensionale Arrays, wird die Nummer des Elements, in das Schreiben von Daten aus dem OpenCL-Puffer beginnt, unter Berücksichtigung der Flagge [AS\\_SERIES](#) ausgerechnet.

Ein Array von zwei oder mehr Dimensionen wird als eindimensional dargestellt. In diesem Fall ist *data\_offset* die Anzahl der Elemente, die in der Darstellung angegeben werden soll, nicht die Anzahl der Elemente in der ersten Dimension.



## CLExecute

Führt OpenCL-Programm. Es gibt drei Varianten der Funktion:

### 1. Lauf der Funktion kernel auf einem einzigen Kern

```
bool CLExecute(
    int          kernel,           // Handle auf den Kernel von OpenCL-Programm
);
```

### 2. Lauf mehrerer Instanzen von kernel (OpenCL-Funktion) mit einer Beschreibung des Problemraums

```
bool CLExecute(
    int          kernel,           // Handle auf den Kernel von OpenCL-Programm
    uint         work_dim,        // Dimension des Raumes von Aufgaben
    const uint&  global_work_offset[], // Anfangsoffset im Raum von Aufgaben
    const uint&  global_work_size[]  // Gesamtzahl der Aufgaben
);
```

### 3. Lauf mehrerer Instanzen von kernel (OpenCL-Funktion) mit einer Beschreibung des Problemraums und der Angabe der Größe des lokalen Teilmenge der Aufgaben in der Gruppe

```
bool CLExecute(
    int          kernel,           // Handle auf den Kernel von OpenCL-Programm
    uint         work_dim,        // Dimension des Raumes von Aufgaben
    const uint&  global_work_offset[], // Anfangsoffset im Raum von Aufgaben
    const uint&  global_work_size[],  // Gesamtzahl der Aufgaben
    const uint&  local_work_size[]   // Anzahl der Aufgaben in der lokalen Gruppe
);
```

#### Parameter

*kernel*

[in] Handle auf den OpenCL-Kernel.

*work\_dim*

[in] Dimension des Raumes von Aufgaben.

*global\_work\_offset[]*

[in] Anfangsoffset im Raum von Aufgaben.

*global\_work\_size[]*

[in] Die Größe einer Teilmenge von Aufgaben.

*local\_work\_size[]*

[in] Die Größe des lokalen Teilmenge der Aufgaben in der Gruppe.

#### Rückgabewert

Gibt bei Erfolg true zurück, ansonsten false. Um Informationen über den Fehler zu erhalten, rufen Sie die Funktion [GetLastError\(\)](#).

#### Hinweis

Betrachten wir das Beispiel der Bedeutung der Parameter:

- *work\_dim* setzt die Dimension des Arrays *work\_items[]*, das die Aufgaben beschreibt. Wenn *work\_dim*=3, dann wird ein 3-dimensionales Array *work\_items[N1, N2, N3]* verwendet.
- *global\_work\_size[]* enthält Werte, die die Größe des Arrays *work\_items[]* angeben. Wenn *work\_dim* = 3, und dementsprechend kann das Array *global\_work\_size[3]* 40 {100, 320} sein. Dann haben wir *work\_items[40, 100, 320]*. Damit die Gesamtanzahl von Aufgaben ist gleich  $40 \times 100 \times 320 = 1\,280\,000$ .
- *local\_work\_size[]* definiert eine Menge der Aufgaben, die der angegebene Kernel des OpenCL-Programms erfüllen wird. Seine Dimension ist der Dimension von *work\_items[]* gleich und erlaubt eine gemeinsame Teilmenge von Aufgaben in kleinere Teilmengen ohne Reste aus der Teilung zu schneiden. In der Tat müssen die Dimensionen des Arrays *local\_work\_size[]* so gewählt werden, dass die globale Gruppe von Aufgaben *work\_items[]* in kleinere Teilmengen geschnitten werden kann. In diesem Beispiel verwenden wir *local\_work\_size[3]={10, 10, 10}*, weil *work\_items[40, 100, 320]* ohne Rest aus dem Array *local\_items[10, 10, 10]* gesamt werden kann.

## Standard Library

This group of chapters contains the technical details of the MQL5 Standard Library and descriptions of all its key components.

MQL5 Standard Library is written in MQL5 and is designed to facilitate writing programs (indicators, scripts, experts) to end users. Library provides convenient access to most of the internal functions MQL5.

MQL5 Standard Library is placed in the working directory of the terminal in the 'Include' folder.

| Section                                                            | Location            |
|--------------------------------------------------------------------|---------------------|
| <a href="#">Base class</a>                                         | Include\            |
| <a href="#">Classes of data</a>                                    | Include\Arrays\     |
| <a href="#">Classes for file operations</a>                        | Include\Files\      |
| <a href="#">Classes for string operations</a>                      | Include\Strings\    |
| <a href="#">Classes for graphic objects</a>                        | Include\Objects\    |
| <a href="#">Class for creating custom graphics</a>                 | Include\Canvas\     |
| <a href="#">Class for working with chart</a>                       | Include\Charts\     |
| <a href="#">Technical indicators</a>                               | Include\Indicators\ |
| <a href="#">Trade classes</a>                                      | Include\Trade\      |
| <a href="#">Trading strategy classes</a>                           | Include\Expert\     |
| <a href="#">Classes for creation of control panels and dialogs</a> | Include\Controls\   |

## Basic Class CObject

Class CObject is the base class for constructing a MQL5 Standard Library .

### Description

Class CObject provides all its descendants to be part of a linked list. Also identifies a number of virtual methods for further implementation in descendant classes.

### Declaration

```
class CObject
```

### Title

```
#include <Object.mqh>
```

### Class Methods

| Attributes                      |                                                      |
|---------------------------------|------------------------------------------------------|
| <a href="#">Prev</a>            | Gets the value of the previous item                  |
| <a href="#">Prev</a>            | Sets the value of the previous item                  |
| <a href="#">Next</a>            | Gets the value of the subsequent element             |
| <a href="#">Next</a>            | Sets the next element                                |
| <b>Compare methods</b>          |                                                      |
| virtual <a href="#">Compare</a> | Returns the result of comparison with another object |
| <b>Input/output</b>             |                                                      |
| virtual <a href="#">Save</a>    | Writes object to a file                              |
| virtual <a href="#">Load</a>    | Reads the object from the file                       |
| virtual <a href="#">Type</a>    | Returns the type of object                           |

Derived classes:

- [CArray](#)
- [CChartObject](#)
- [CChart](#)
- [CString](#)
- [CFile](#)
- [CList](#)
- [CTreeNode](#)

## Prev

Gets a pointer to the previous list item.

```
CObject* Prev()
```

### Return Value

Pointer to the previous list item. If an item is listed first, then return NULL.

### Example:

```
//--- example for CObject::Prev()
#include <Object.mqh>
//---
void OnStart()
{
    CObject *object_first,*object_second;
    //---
    object_first=new CObject;
    if(object_first==NULL)
    {
        printf("Object create error");
        return;
    }
    object_second=new CObject;
    if(object_second==NULL)
    {
        printf("Object create error");
        delete object_first;
        return;
    }
    //--- set interconnect
    object_first.Next(object_second);
    object_second.Prev(object_first);
    //--- use prev object
    CObject *object=object_second.Prev();
    //--- delete objects
    delete object_first;
    delete object_second;
}
```

## Prev

Sets the pointer to the previous list item.

```
void Prev(  
    CObject* object    // Pointer to the previous list item  
)
```

### Parameters

*object*

[in] New value pointer to the previous list item.

### Example:

```
///--- example for CObject::Prev(CObject*)  
#include <Object.mqh>  
///---  
void OnStart()  
{  
    CObject *object_first,*object_second;  
    ///---  
    object_first=new CObject;  
    if(object_first==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    object_second=new CObject;  
    if(object_second==NULL)  
    {  
        printf("Object create error");  
        delete object_first;  
        return;  
    }  
    ///--- set interconnect  
    object_first.Next(object_second);  
    object_second.Prev(object_first);  
    ///--- use objects  
    ///--- ...  
    ///--- delete objects  
    delete object_first;  
    delete object_second;  
}
```

## Next

Gets a pointer to the next element of the list.

```
CObject* Next()
```

### Return Value

Pointer to the next item in the list. If the last item in the list, returns NULL.

### Example:

```
//--- example for CObject::Next()
#include <Object.mqh>
//---
void OnStart()
{
    CObject *object_first,*object_second;
    //---
    object_first=new CObject;
    if(object_first==NULL)
    {
        printf("Object create error");
        return;
    }
    object_second=new CObject;
    if(object_second==NULL)
    {
        printf("Object create error");
        delete object_first;
        return;
    }
    //--- set interconnect
    object_first.Next(object_second);
    object_second.Prev(object_first);
    //--- use next object
    CObject *object=object_first.Next();
    //--- delete objects
    delete object_first;
    delete object_second;
}
```

## Next

Sets the pointer to the next element of the list.

```
void Next(  
    CObject* object    // Pointer to the next list item  
)
```

### Parameters

*object*

[in] New value pointer to the next list item.

### Example:

```
///--- example for CObject::Next(CObject*)  
#include <Object.mqh>  
///---  
void OnStart()  
{  
    CObject *object_first,*object_second;  
    ///---  
    object_first=new CObject;  
    if(object_first==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    object_second=new CObject;  
    if(object_second==NULL)  
    {  
        printf("Object create error");  
        delete object_first;  
        return;  
    }  
    ///--- set interconnect  
    object_first.Next(object_second);  
    object_second.Prev(object_first);  
    ///--- use objects  
    ///--- ...  
    ///--- delete objects  
    delete object_first;  
    delete object_second;  
}
```



## Compare

Compares the data item in the list with data on another element of the list.

```
virtual int Compare(  
    CObject const * node,      // Node to compare with  
    int mode=0                // Compare mode  
) const
```

### Parameters

*node*

[in] Pointer to a list item to compare

*mode=0*

[in] Variant Comparison

### Return Value

0 - in case the list items are equal, -1 - if the list item is less than the item in the list for comparison (node), 1 - if the list item more than item in the list for comparison (node).

### Note

Method Compare () in class CObject always returns 0 and does not perform any action. If you want to compare data derived class, the method Compare (...) Should be implemented. The mode parameter should be used when implementing multivariate comparison.

### Example:

```
//--- example for CObject::Compare(...)  
#include <Object.mqh>  
//---  
void OnStart()  
{  
    CObject *object_first,*object_second;  
    //---  
    object_first=new CObject;  
    if(object_first==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    object_second=new CObject;  
    if(object_second==NULL)  
    {  
        printf("Object create error");  
        delete object_first;  
        return;  
    }  
    //--- set interconnect  
    object_first.Next(object_second);
```

```
object_second.Prev(object_first);  
//--- compare objects  
int result=object_first.Compare(object_second);  
//--- delete objects  
delete object_first;  
delete object_second;  
}
```

## Save

Saves data element list in the file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] Handle to open earlier, with the function `FileOpen()`, binary

### Return Value

true - if successfully completed, false - if an error.

### Note

Method `Save(int)` in class `CObject` always returns true and does not perform any action. If you want to save the data derived class in the file, the method `Save(int)` should be implemented.

### Example:

```
//--- example for CObject::Save(int)  
#include <Object.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CObject *object=new CObject;  
    //---  
    if(object!=NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- set objects data  
    //--- . . .  
    //--- open file  
    file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!object.Save(file_handle))  
        {  
            //--- file save error  
            printf("File save: Error %d!",GetLastError());  
            delete object;  
            FileClose(file_handle);  
            //---  
            return;  
        }  
    }  
}
```

```
        FileClose(file_handle);  
    }  
    delete object;  
}
```

## Load

Loads data item in the list from a file.

```
virtual bool Load(  
    int file_handle    // handle to file  
)
```

### Parameters

*file\_handle*

[in] Handle to open earlier, with the function `FileOpen()`, binary

### Return Value

true - if successfully completed, false - if an error.

### Note

Method `Load(int)` in class `CObject` always returns true and does not perform any action. If you want to load the data derived class from a file, the method `Load(int)` should be implemented.

### Example:

```
//--- example for CObject::Load(int)  
#include <Object.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CObject *object=new CObject;  
    //---  
    if(object!=NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- open file  
    file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!object.Load(file_handle))  
        {  
            //--- file load error  
            printf("File load: Error %d!",GetLastError());  
            delete object;  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }  
}
```

```
//--- use object  
//--- . . .  
delete object;  
}
```

## Type

Gets the type identifier.

```
virtual int Type() const
```

### Return Value

Type identifier (for CObject - 0).

### Example:

```
//--- example for CObject::Type()
#include <Object.mqh>
//---
void OnStart()
{
    CObject *object=new CObject;
    //---
    object=new CObject;
    if(object ==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- get objects type
    int type=object.Type();
    //--- delete object
    delete object;
}
```

## Data Structures

This section contains the technical details of working with various data structures (arrays, linked lists, etc.) and description of the relevant components of the MQL5 Standard Library .

Using classes of data structures, will save time when creating custom data stores of various formats (including composite data structures).

MQL5 Standard Library (in terms of data sets) is placed in the working directory of the terminal in the Include\Arrays folder.

## Data Arrays

Use of classes of dynamic arrays of data will save time when creating a custom data stores of various formats (including multidimensional arrays).

MQL5 Standard Library (in terms of arrays of data) is placed in the working directory of the terminal in the Include\Arrays folder.

| Class                                                          | Description                                                                                                                   |
|----------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------|
| <a href="#"><u>Base class of dynamic data array CArray</u></a> | Base class of dynamic data array                                                                                              |
| <a href="#"><u>CArrayChar</u></a>                              | Dynamic array of variables of type char or uchar                                                                              |
| <a href="#"><u>CArrayShort</u></a>                             | Dynamic array of variables of type short or ushort                                                                            |
| <a href="#"><u>CArrayInt</u></a>                               | Dynamic array of variables of type int or uint                                                                                |
| <a href="#"><u>CArrayLong</u></a>                              | Dynamic array of variables of type long or ulong                                                                              |
| <a href="#"><u>CArrayFloat</u></a>                             | Dynamic array of variables of type float                                                                                      |
| <a href="#"><u>CArrayDouble</u></a>                            | Dynamic array of variables of type double                                                                                     |
| <a href="#"><u>CArrayString</u></a>                            | Dynamic array of variables of type string                                                                                     |
| <a href="#"><u>Base class of object array CArrayObj</u></a>    | Dynamic array of pointers <a href="#"><u>CObject</u></a>                                                                      |
| <a href="#"><u>Base class of list CList</u></a>                | Provides the ability to work with a list of instances of <a href="#"><u>CObject</u></a> and its descendant                    |
| <a href="#"><u>CTreeNode</u></a>                               | Provides the ability to work with nodes of the binary tree <a href="#"><u>CTree</u></a>                                       |
| <a href="#"><u>CTree</u></a>                                   | Provides the ability to work with the binary tree of the <a href="#"><u>CTreeNode</u></a> class instances and its descendants |



# CArray

CArray class is the base class a dynamic array of variables.

## Description

Class CArray provides the ability to work with a dynamic array of variables in the part of the attributes of management of memory allocation, sorting, and working with files.

## Declaration

```
class CArray : public CObject
```

## Title

```
#include <Arrays\Array.mqh>
```

## Class Methods

| Attributes                   |                                                                                                  |
|------------------------------|--------------------------------------------------------------------------------------------------|
| <a href="#">Step</a>         | Gets the step increment size of the array                                                        |
| <a href="#">Step</a>         | Set the increment size of the array                                                              |
| <a href="#">Total</a>        | Gets the number of elements in the array                                                         |
| <a href="#">Available</a>    | Gets the number of free elements of the array are available without additional memory allocation |
| <a href="#">Max</a>          | Gets the maximum possible size of the array without memory reallocation                          |
| <a href="#">IsSorted</a>     | Gets sign sorted array to the specified option                                                   |
| <a href="#">SortMode</a>     | Gets the version of the sorting array                                                            |
| Clear methods                |                                                                                                  |
| <a href="#">Clear</a>        | Deletes all of the array elements without memory release                                         |
| Sort methods                 |                                                                                                  |
| <a href="#">Sort</a>         | Sorts an array to the specified option                                                           |
| Input/output                 |                                                                                                  |
| virtual <a href="#">Save</a> | Saves data array in the file                                                                     |
| virtual <a href="#">Load</a> | Loads data array from a file                                                                     |

Derived classes:

- [CArrayChar](#)

- [CArrayShort](#)
- [CArrayInt](#)
- [CArrayLong](#)
- [CArrayFloat](#)
- [CArrayDouble](#)
- [CArrayString](#)
- [CArrayObj](#)

## Step

Gets the step increment size of the array.

```
int Step() const
```

### Return Value

Increment size of the array.

### Example:

```
//--- example for CArray::Step()
#include <Arrays\Array.mqh>
//---
void OnStart()
{
    CArray *array=new CArray;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- get resize step
    int step=array.Step();
    //--- use array
    //--- ...
    //--- delete array
    delete array;
}
```

## Step

Sets the increment size of the array.

```
bool Step(  
    int step    // step  
)
```

### Parameters

*step*

[in] The new value of step increments in the size of the array.

### Return Value

true if successful, false - if there was an attempt to establish a step less than or equal to zero.

### Example:

```
///--- example for CArray::Step(int)  
#include <Arrays\Array.mqh>  
///---  
void OnStart()  
{  
    CArray *array=new CArray;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- set resize step  
    bool result=array.Step(1024);  
    ///--- use array  
    ///--- ...  
    ///--- delete array  
    delete array;  
}
```

## Total

Gets the number of elements in the array.

```
int Total() const;
```

### Return Value

Number of elements in the array.

### Example:

```
//--- example for CArray::Total()
#include <Arrays\Array.mqh>
//---
void OnStart()
{
    CArray *array=new CArray;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- check total
    int total=array.Total();
    //--- use array
    //--- ...
    //--- delete array
    delete array;
}
```

## Available

Gets the number of free elements of the array are available without additional memory allocation.

```
int Available() const
```

### Return Value

Number of free elements of the array are available without additional memory allocation.

### Example:

```
//--- example for CArray::Available()
#include <Arrays\Array.mqh>
//---
void OnStart()
{
    CArray *array=new CArray;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- check available
    int available=array.Available();
    //--- use array
    //--- ...
    //--- delete array
    delete array;
}
```

## Max

Gets the maximum possible size of the array without memory reallocation.

```
int Max() const
```

### Return Value

The maximum possible size of the array without reallocation memory.

### Example:

```
//--- example for CArray::Max()
#include <Arrays\Array.mqh>
//---
void OnStart()
{
    CArray *array=new CArray;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- check maximum size
    int max=array.Max();
    //--- use array
    //--- ...
    //--- delete array
    delete array;
}
```

## IsSorted

Gets sign sorted array to the specified option.

```
bool IsSorted(  
    int mode=0      // Sorting mode  
) const
```

### Parameters

*mode=0*

[in] Tested version of the sort.

### Return Value

Flag of the sorted list. If the list is sorted by the specified version - true, otherwise - false.

### Note

Symptom sorted array can not be changed directly. Symptom sorted set method Sort () and reset any methods to add / insert except InsertSort (...).

### Example:

```
//--- example for CArray::IsSorted()  
#include <Arrays\Array.mqh>  
//---  
void OnStart()  
{  
    CArray *array=new CArray;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- check sorted  
    if(array.IsSorted())  
    {  
        //--- use methods for sorted array  
        //--- ...  
    }  
    //--- delete array  
    delete array;  
}
```



## SortMode

Gets the version of the sorting array.

```
int SortMode() const;
```

### Return Value

Sorting mode.

### Example:

```
//--- example for CArray::SortMode()
#include <Arrays\Array.mqh>
//---
void OnStart()
{
    CArray *array=new CArray;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- check sort mode
    int sort_mode=array.SortMode();
    //--- use array
    //--- ...
    //--- delete array
    delete array;
}
```

## Clear

Deletes all of the array elements without memory release.

```
void Clear()
```

### Return Value

None.

### Example:

```
//--- example for CArray::Clear()
#include <Arrays\Array.mqh>
//---
void OnStart()
{
    CArray *array=new CArray;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- use array
    //--- ...
    //--- clear array
    array.Clear();
    //--- delete array
    delete array;
}
```

## Sort

Sorts an array to the specified option.

```
void Sort(  
    int mode=0      // Sorting mode  
)
```

### Parameters

*mode=0*

[in] Mode of array sorting.

### Return Value

No.

### Note

Sorting an array is always ascending. For arrays of primitive data types (CArrayChar, CArrayShort, etc.), the parameter mode is not used. For the array CArrayObj, multivariate sort should be implemented in the method Sort (int) derived class.

### Example:

```
//--- example for CArray::Sort(int)  
#include <Arrays\Array.mqh>  
//---  
void OnStart()  
{  
    CArray *array=new CArray;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- sorting by mode 0  
    array.Sort(0);  
    //--- use array  
    //--- ...  
    //--- delete array  
    delete array;  
}
```

## Save

Saves data array in the file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] Handle to open earlier, with the function `FileOpen (...)`, binary file.

### Return Value

true - if successfully completed, false - if an error.

### Example:

```
//--- example for CArray::Save(int)  
#include <Arrays\Array.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CArray *array=new CArray;  
    //---  
    if(array!=NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- open file  
    file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!array.Save(file_handle))  
        {  
            //--- file save error  
            printf("File save: Error %d!",GetLastError());  
            delete array;  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }  
    //--- delete array  
    delete array;  
}
```

## Load

Loads data array from a file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] Handle to open earlier, with the function FileOpen (...), binary file.

### Return Value

true - if successfully completed, false - if an error.

### Example:

```
//--- example for CArray::Load(...)  
#include <Arrays\Array.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CArray *array=new CArray;  
    //---  
    if(array!=NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- open file  
    file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!array.Load(file_handle))  
        {  
            //--- file load error  
            printf("File load: Error %d!",GetLastError());  
            delete array;  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }  
    //--- delete array  
    delete array;  
}
```

## CArrayChar

CArrayChar class is a class of dynamic array of variables of type char or uchar.

### Description

Class CArrayChar provides the ability to work with a dynamic array of variables of type char or uchar. In the class implemented the ability to add / insert / delete elements in an array, sort array, searching in sorted array. In addition, the implemented methods of work with the file.

### Declaration

```
class CArrayChar : public CArray
```

### Title

```
#include <Arrays\ArrayChar.mqh>
```

### Class Methods

| Memory control              |                                                                             |
|-----------------------------|-----------------------------------------------------------------------------|
| <a href="#">Reserve</a>     | Allocates memory to increase the size of the array                          |
| <a href="#">Resize</a>      | Sets a new (smaller) size of the array                                      |
| <a href="#">Shutdown</a>    | Clears the array with a full memory release                                 |
| Add methods                 |                                                                             |
| <a href="#">Add</a>         | Adds an element to the end of the array                                     |
| <a href="#">AddArray</a>    | Adds to the end of the array elements from another array                    |
| <a href="#">AddArray</a>    | Adds to the end of the array elements from another array                    |
| <a href="#">Insert</a>      | Inserts an element in the array to the specified position                   |
| <a href="#">InsertArray</a> | Inserts an array of elements from another array with the specified position |
| <a href="#">InsertArray</a> | Inserts an array of elements from another array with the specified position |
| <a href="#">AssignArray</a> | Copies the array elements from another array                                |
| <a href="#">AssignArray</a> | Copies the array elements from another array                                |
| Modify methods              |                                                                             |
| <a href="#">Update</a>      | Changes the element at the specified position array                         |

|                                    |                                                                            |
|------------------------------------|----------------------------------------------------------------------------|
| <a href="#">Shift</a>              | Moves an item from a given position in the array to the specified offset   |
| <b>Delete methods</b>              |                                                                            |
| <a href="#">Delete</a>             | Removes the element from the specified position array                      |
| <a href="#">DeleteRange</a>        | Deletes a group of elements from the specified position array              |
| <b>Access methods</b>              |                                                                            |
| <a href="#">At</a>                 | Gets the element from the specified position array                         |
| <b>Compare methods</b>             |                                                                            |
| <a href="#">CompareArray</a>       | Compares array with another array                                          |
| <a href="#">CompareArray</a>       | Compares array with another array                                          |
| <b>Sorted array methods</b>        |                                                                            |
| <a href="#">InsertSort</a>         | Inserts element in a sorted array                                          |
| <a href="#">Search</a>             | Searches for an element equal to the sample in the sorted array            |
| <a href="#">SearchGreat</a>        | Searches for an element of more samples in sorted array                    |
| <a href="#">SearchLess</a>         | Searches for an element less than the sample in the sorted array           |
| <a href="#">SearchGreatOrEqual</a> | Searches for an element greater than or equal to the model in sorted array |
| <a href="#">SearchLessOrEqual</a>  | Searches for an element less than or equal to the model in sorted array    |
| <a href="#">SearchFirst</a>        | Finds the first element equal to the model in sorted array                 |
| <a href="#">SearchLast</a>         | Finds the last element equal to the model in sorted array                  |
| <a href="#">SearchLinear</a>       | Searches for the element equal to the sample in the array                  |
| <b>Input/output</b>                |                                                                            |
| virtual <a href="#">Save</a>       | Saves data array in the file                                               |
| virtual <a href="#">Load</a>       | Loads data array from a file                                               |
| virtual <a href="#">Type</a>       | Gets the type identifier of the array                                      |

## Reserve

Allocates memory to increase the size of the array.

```
bool Reserve (  
    int size      // Number  
)
```

### Parameters

*size*

[in] The number of additional elements of the array.

### Return Value

true if successful, false - if there was an attempt to seek the amount is less than or equal to zero, or if the array did not increase.

### Note

To reduce memory fragmentation, increase the size of the array is made with a step previously given through the method of Step (int), or 16 (default).

### Example:

```
///--- example for CArrayChar::Reserve(int)  
#include <Arrays\ArrayChar.mqh>  
///---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- reserve memory  
    if(!array.Reserve(1024))  
    {  
        printf("Reserve error");  
        delete array;  
        return;  
    }  
    ///--- use array  
    ///--- . . .  
    ///--- delete array  
    delete array;  
}
```



## Resize

Sets a new (smaller) size of the array.

```
bool Resize(  
    int size      // Size  
)
```

### Parameters

*size*

[in] New size of the array.

### Return Value

true if successful, false - if there was an attempt to set the size of less than or equal to zero.

### Note

Changing the size of the array allows optimal use of memory. Superfluous elements on the right lost. To reduce fragmentation of memory, change the size of the array is made with a step previously given through the method of Step (int), or 16 (default).

### Example:

```
//--- example for CArrayChar::Resize(int)  
#include <Arrays\ArrayChar.mqh>  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- resize array  
    if(!array.Resize(10))  
    {  
        printf("Resize error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

## Shutdown

Clears the array with a full memory release.

```
bool Shutdown()
```

### Return Value

true if successful, false - if an error occurred.

### Example:

```
//--- example for CArrayChar::Shutdown()
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
{
    CArrayChar *array=new CArrayChar;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- add arrays elements
    //--- . . .
    //--- shutdown array
    if(!array.Shutdown())
    {
        printf("Shutdown error");
        delete array;
        return;
    }
    //--- delete array
    delete array;
}
```

## Add

Adds an element to the end of the array.

```
bool Add(  
    char element    // Element to add  
)
```

### Parameters

*element*

[in] value of the element to add to the array.

### Return Value

true if successful, false - if you can not add an element.

### Example:

```
///--- example for CArrayChar::Add(char)  
#include <Arrays\ArrayChar.mqh>  
///---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    for(int i=0;i<100;i++)  
    {  
        if(!array.Add(i))  
        {  
            printf("Element addition error");  
            delete array;  
            return;  
        }  
    }  
    ///--- use array  
    ///--- . . .  
    ///--- delete array  
    delete array;  
}
```

## AddArray

Adds to the end of the array elements from another array.

```
bool AddArray(  
    const char& src[]      // Source array  
)
```

### Parameters

*src[]*

[in] Reference to an array of source elements to add.

### Return Value

true if successful, false - if you can not add items.

### Example:

```
//--- example for CArrayChar::AddArray(const char &[])  
#include <Arrays\ArrayChar.mqh>  
//---  
char src[];  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add another array  
    if(!array.AddArray(src))  
    {  
        printf("Array addition error");  
        delete array;  
        return;  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

## AddArray

Adds to the end of the array elements from another array.

```
bool AddArray(  
    const CArrayChar* src      // Pointer to the source  
)
```

### Parameters

*src*

[in] Pointer to an instance of class CArrayChar-source elements to add.

### Return Value

true if successful, false - if you can not add items.

### Example:

```
///--- example for CArrayChar::AddArray(const CArrayChar*)  
#include <Arrays\ArrayChar.mqh>  
///---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- create source array  
    CArrayChar *src=new CArrayChar;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    ///--- add source arrays elements  
    ///--- . . .  
    ///--- add another array  
    if(!array.AddArray(src))  
    {  
        printf("Array addition error");  
        delete src;  
        delete array;  
        return;  
    }  
    ///--- delete source array  
    delete src;
```

```
//--- use array  
//--- . . .  
//--- delete array  
delete array;  
}
```

## Insert

Inserts an element in the array to the specified position.

```
bool Insert(  
    char element,    // Element to insert  
    int pos          // Position  
)
```

### Parameters

*element*

[in] Value of the element to be inserted into an array

*pos*

[in] Position in the array to insert

### Return Value

true if successful, false - if you can not insert the element.

### Example:

```
//--- example for CArrayChar::Insert(char,int)  
#include <Arrays\ArrayChar.mqh>  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- insert elements  
    for(int i=0;i<100;i++)  
    {  
        if(!array.Insert(i,0))  
        {  
            printf("Insert error");  
            delete array;  
            return;  
        }  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

## InsertArray

Inserts an array of elements from another array with the specified position.

```
bool  InsertArray(  
    const char&  src[],      // Source array  
    int          pos         // Position  
)
```

### Parameters

*src[]*

[in] Reference to an array of source elements to insert

*pos*

[in] Position in the array to insert

### Return Value

true if successful, false - if you can not paste items.

### Example:

```
//--- example for CArrayChar::InsertArray(const char &[],int)  
#include <Arrays\ArrayChar.mqh>  
//---  
char src[];  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- insert another array  
    if(!array.InsertArray(src,0))  
    {  
        printf("Array inserting error");  
        delete array;  
        return;  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```



## InsertArray

Inserts an array of elements from another array with the specified position.

```
bool InsertArray(  
    CArrayChar* src,      // Pointer to the source  
    int pos              // Position  
)
```

### Parameters

*src*

[in] Pointer to an instance of class CArrayChar-source elements to insert.

*pos*

[in] Position in the array to insert

### Return Value

true if successful, false - if you can not paste items.

### Example:

```
//--- example for CArrayChar::InsertArray(const CArrayChar*,int)  
#include <Arrays\ArrayChar.mqh>  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source array  
    CArrayChar *src=new CArrayChar;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    //--- add source arrays elements  
    //--- . . .  
    //--- insert another array  
    if(!array.InsertArray(src,0))  
    {  
        printf("Array inserting error");  
        delete src;  
        delete array;  
    }  
}
```

```
        return;  
    }  
    //--- delete source array  
    delete src;  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

## AssignArray

Copies the array elements from another array.

```
bool AssignArray(  
    const char& src[]      // Source array  
)
```

### Parameters

*src[]*

[in] Reference to an array of source elements to copy.

### Return Value

true if successful, false - if you can not copy the items.

### Example:

```
///--- example for CArrayChar::AssignArray(const char &[])  
#include <Arrays\ArrayChar.mqh>  
///---  
char src[];  
///---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- assign another array  
    if(!array.AssignArray(src))  
    {  
        printf("Array assigned error");  
        delete array;  
        return;  
    }  
    ///--- use array  
    ///--- . . .  
    ///--- delete array  
    delete array;  
}
```

## AssignArray

Copies the array elements from another array.

```
bool AssignArray(  
    const CArrayChar* src      // Pointer to the source  
)
```

### Parameters

*src*

[in] Pointer to an instance of class CArrayChar-source element to copy.

### Return Value

true if successful, false - if you can not copy the items.

### Example:

```
///--- example for CArrayChar::AssignArray(const CArrayChar*)  
#include <Arrays\ArrayChar.mqh>  
///---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- create source array  
    CArrayChar *src =new CArrayChar;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    ///--- add source arrays elements  
    ///--- . . .  
    ///--- assign another array  
    if(!array.AssignArray(src))  
    {  
        printf("Array assigned error");  
        delete src;  
        delete array;  
        return;  
    }  
    ///--- arrays is identical  
    ///--- delete source array
```

```
delete src;  
//--- use array  
//--- . . .  
//--- delete array  
delete array;  
}
```

## Update

Changes the element at the specified position array.

```
bool Update(  
    int    pos,           // Position  
    char   element       // Value  
)
```

### Parameters

*pos*

[in] Position of the element in the array to change

*element*

[in] New value element

### Return Value

true if successful, false - if you can not change the element.

### Example:

```
//--- example for CArrayChar::Update(int,char)  
#include <Arrays\ArrayChar.mqh>  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- update element  
    if(!array.Update(0,'A'))  
    {  
        printf("Update error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

## Shift

Moves an item from a given position in the array to the specified offset.

```
bool Shift(  
    int pos,          // Position  
    int shift         // Value  
)
```

### Parameters

*pos*

[in] Position of the moved element array

*shift*

[in] The value of displacement (both positive and negative).

### Return Value

true if successful, false - if you can not move the item.

### Example:

```
//--- example for CArrayChar::Shift(int,int)  
#include <Arrays\ArrayChar.mqh>  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- shift element  
    if(!array.Shift(10,-5))  
    {  
        printf("Shift error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

## Delete

Removes the element from the given position in the array.

```
bool Delete(  
    int pos    // Position  
)
```

### Parameters

*pos*

[in] Position removes the element in the array.

### Return Value

true if successful, false - if you can not remove the element.

### Example:

```
//--- example for CArrayChar::Delete(int)  
#include <Arrays\ArrayChar.mqh>  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- delete element  
    if(!array.Delete(0))  
    {  
        printf("Delete error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```



## DeleteRange

Deletes a group of elements from a given position in the array.

```
bool DeleteRange(  
    int from,      // Position of the first element  
    int to         // Positions of the last element  
)
```

### Parameters

*from*

[in] Position of the first removes the element in the array.

*to*

[in] Position of the last deleted element in the array.

### Return Value

true if successful, false - if you can not remove elements.

### Example:

```
//--- example for CArrayChar::DeleteRange(int,int)  
#include <Arrays\ArrayChar.mqh>  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- delete elements  
    if(!array.DeleteRange(0,10))  
    {  
        printf("Delete error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

## At

Gets the element from the given position in the array.

```
char At(  
    int pos    // Position  
) const
```

### Parameters

*pos*

[in] Position of the desired element in the array.

### Return Value

The value of the element in case of success, CHAR\_MAX-if there was an attempt to get an element of not existing positions (the last error ERR\_OUT\_OF\_RANGE).

### Note

Of course, CHAR\_MAX may be a valid value of an array element, so having a value, always check the last error code.

### Example:

```
//--- example for CArrayChar::At(int)  
#include <Arrays\ArrayChar.mqh>  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    for(int i=0;i<array.Total();i++)  
    {  
        char result=array.At(i);  
        if(result==CHAR_MAX && GetLastError()==ERR_OUT_OF_RANGE)  
        {  
            //--- error of reading from array  
            printf("Get element error");  
            delete array;  
            return;  
        }  
        //--- use element  
        //--- . . .  
    }  
}
```

```
//--- delete array  
delete array;  
}
```

## CompareArray

Compares array with another array.

```
bool CompareArray(  
    const char& src[]      // Source array  
    ) const
```

### Parameters

*src[]*

[in] Reference to an array of source elements for comparison.

### Return Value

true if arrays are equal, false - if not.

### Example:

```
///--- example for CArrayChar::CompareArray(const char &[])  
#include <Arrays\ArrayChar.mqh>  
///---  
char src[];  
///---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- compare with another array  
    int result=array.CompareArray(src);  
    ///--- delete array  
    delete array;  
}
```

## CompareArray

Compares array with another array.

```
bool CompareArray(  
    const CArrayChar* src      // Pointer to the sources  
    ) const
```

### Parameters

*src*

[in] Pointer to an instance of class CArrayChar-source elements for comparison.

### Return Value

true if arrays are equal, false - if not.

### Example:

```
//--- example for CArrayChar::CompareArray(const CArrayChar*)  
#include <Arrays\ArrayChar.mqh>  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source array  
    CArrayChar *src=new CArrayChar;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    //--- add source arrays elements  
    //--- . . .  
    //--- compare with another array  
    int result=array.CompareArray(src);  
    //--- delete arrays  
    delete src;  
    delete array;  
}
```

## InsertSort

Inserts element in a sorted array.

```
bool InsertSort(  
    char element    // Element to insert  
)
```

### Parameters

*element*

[in] Value of the element to be inserted into a sorted array

### Return Value

true if successful, false - if you can not insert the element.

### Example:

```
///--- example for CArrayChar::InsertSort(char)  
#include <Arrays\ArrayChar.mqh>  
///---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    ///--- sort array  
    array.Sort();  
    ///--- insert element  
    if(!array.InsertSort('A'))  
    {  
        printf("Insert error");  
        delete array;  
        return;  
    }  
    ///--- delete array  
    delete array;  
}
```

## Search

Searches for an element equal to the sample in the sorted array.

```
int Search(  
    char element    // Sample  
    ) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayChar::Search(char)  
#include <Arrays\ArrayChar.mqh>  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.Search('A')!=-1) printf("Element found");  
    else                      printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchGreat

Searches for an element of more samples in sorted array.

```
int SearchGreat(  
    char element    // Sample  
    ) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayChar::SearchGreat(char)  
#include <Arrays\ArrayChar.mqh>  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchGreat('A')!=-1) printf("Element found");  
    else                          printf("Element not found");  
    //--- delete array  
    delete array;  
}
```



## SearchLess

Searches for an element less than the sample in the sorted array.

```
int SearchLess(  
    char element    // Sample  
    ) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayChar::SearchLess(char)  
#include <Arrays\ArrayChar.mqh>  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchLess('A')!=-1) printf("Element found");  
    else                          printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchGreatOrEqual

Searches for an element greater than or equal to the sample in the sorted array

```
int SearchGreatOrEqual(  
    char element    // Sample  
) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayChar::SearchGreatOrEqual(char)  
#include <Arrays\ArrayChar.mqh>  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchGreatOrEqual('A')!=-1) printf("Element found");  
    else                                printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchLessOrEqual

Searches for an element less than or equal to the sample in the sorted array.

```
int SearchLessOrEqual(  
    char element    // Sample  
    ) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayChar::SearchLessOrEqual(char)  
#include <Arrays\ArrayChar.mqh>  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchLessOrEqual('A')!=-1) printf("Element found");  
    else                                printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchFirst

Finds the first element equal to the sample in the sorted array.

```
int SearchFirst(  
    char element    // Sample  
    ) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayChar::SearchFirst(char)  
#include <Arrays\ArrayChar.mqh>  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchFirst('A')!=-1) printf("Element found");  
    else                          printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchLast

Finds the last element equal to the model in sorted array.

```
int SearchLast(  
    char element    // Sample  
    ) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayChar::SearchLast(char)  
#include <Arrays\ArrayChar.mqh>  
//---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchLast('A')!=-1) printf("Element found");  
    else                          printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchLinear

Searches for the element equal to the sample in the array.

```
int SearchLinear(  
    char element    // Sample  
) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the element was not found.

### Note

The method uses the linear search (or sequential search) algorithm for unsorted arrays.

### Example:

```
///--- example for CArrayChar::SearchLinear(char)  
#include <Arrays\ArrayChar.mqh>  
///---  
void OnStart()  
{  
    CArrayChar *array=new CArrayChar;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    ///--- search element  
    if(array.SearchLinear('A')!=-1) printf("Element found");  
    else                             printf("Element not found");  
    ///--- delete array  
    delete array;  
}
```

## Save

Saves data array in the file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] Handle to open earlier, with the function `FileOpen (...)`, binary file.

### Return Value

true - if successfully completed, false - if an error.

### Example:

```
//--- example for CArrayChar::Save(int)  
#include <Arrays\ArrayChar.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array!=NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- open file  
    file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!array.Save(file_handle))  
        {  
            //--- file save error  
            printf("File save: Error %d!",GetLastError());  
            delete array;  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }  
    //--- delete array  
    delete array;  
}
```

## Load

Loads data array from a file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] Handle to open earlier, with the function FileOpen (...), binary file.

### Return Value

true - if successfully completed, false - if an error.

### Example:

```
//--- example for CArrayChar::Load(int)  
#include <Arrays\ArrayChar.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CArrayChar *array=new CArrayChar;  
    //---  
    if(array!=NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- open file  
    file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!array.Load(file_handle))  
        {  
            //--- file load error  
            printf("File load: Error %d!",GetLastError());  
            delete array;  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }  
    //--- use arrays elements  
    for(int i=0;i<array.Total();i++)  
    {  
        printf("Element[%d] = '%c'",i,array.At(i));  
    }  
}
```



```
    }  
    ///--- delete array  
    delete array;  
}
```

## Type

Gets the type identifier of the array.

```
virtual int Type() const
```

### Return Value

ID type of the array (for CArrayChar - 77).

### Example:

```
//--- example for CArrayChar::Type()
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
{
    CArrayChar *array=new CArrayChar;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- get array type
    int type=array.Type();
    //--- delete array
    delete array;
}
```

## CArrayShort

CArrayShort class is a class of dynamic array of variables of type short or ushort.

### Description

Class CArrayShort provides the ability to work with a dynamic array of variables of type short or ushort. In the class implemented the ability to add / insert / delete elements in an array, sort array, searching in sorted array. In addition, the implemented methods of work with the file.

### Declaration

```
class CArrayShort : public CArray
```

### Title

```
#include <Arrays\ArrayShort.mqh>
```

### Class Methods

| Memory control              |                                                                             |
|-----------------------------|-----------------------------------------------------------------------------|
| <a href="#">Reserve</a>     | Allocates memory to increase the size of the array                          |
| <a href="#">Resize</a>      | Sets a new (smaller) size of the array                                      |
| <a href="#">Shutdown</a>    | Clears the array with a full memory release                                 |
| <b>Add methods</b>          |                                                                             |
| <a href="#">Add</a>         | Adds an element to the end of the array                                     |
| <a href="#">AddArray</a>    | Adds to the end of the array elements from another array                    |
| <a href="#">AddArray</a>    | Adds to the end of the array elements from another array                    |
| <a href="#">Insert</a>      | Inserts an element in the array to the specified position                   |
| <a href="#">InsertArray</a> | Inserts an array of elements from another array with the specified position |
| <a href="#">InsertArray</a> | Inserts an array of elements from another array with the specified position |
| <a href="#">AssignArray</a> | Copies the array elements from another array                                |
| <a href="#">AssignArray</a> | Copies the array elements from another array                                |
| <b>Update methods</b>       |                                                                             |
| <a href="#">Update</a>      | Changes the element at the specified position array                         |

|                                    |                                                                            |
|------------------------------------|----------------------------------------------------------------------------|
| <a href="#">Shift</a>              | Moves an item from a given position in the array to the specified offset   |
| <b>Delete methods</b>              |                                                                            |
| <a href="#">Delete</a>             | Removes the element from the specified position array                      |
| <a href="#">DeleteRange</a>        | Deletes a group of elements from the specified position array              |
| <b>Access methods</b>              |                                                                            |
| <a href="#">At</a>                 | Gets the element from the specified position array                         |
| <b>Compare methods</b>             |                                                                            |
| <a href="#">CompareArray</a>       | Compares array with another array                                          |
| <a href="#">CompareArray</a>       | Compares array with another array                                          |
| <b>Sorted array operations</b>     |                                                                            |
| <a href="#">InsertSort</a>         | Inserts element in a sorted array                                          |
| <a href="#">Search</a>             | Searches for an element equal to the model in sorted array                 |
| <a href="#">SearchGreat</a>        | Searches for an element of more samples in sorted array                    |
| <a href="#">SearchLess</a>         | Searches for an element less than the sample in the sorted array           |
| <a href="#">SearchGreatOrEqual</a> | Searches for an element greater than or equal to the model in sorted array |
| <a href="#">SearchLessOrEqual</a>  | Searches for an element less than or equal to the model in sorted array    |
| <a href="#">SearchFirst</a>        | Finds the first element equal to the model in sorted array                 |
| <a href="#">SearchLast</a>         | Finds the last element equal to the model in sorted array                  |
| <a href="#">SearchLinear</a>       | Searches for the element equal to the sample in the array                  |
| <b>Input/output</b>                |                                                                            |
| virtual <a href="#">Save</a>       | Saves data array in the file                                               |
| virtual <a href="#">Load</a>       | Loads data array from a file                                               |
| virtual <a href="#">Type</a>       | Gets the type identifier of the array                                      |

## Reserve

Allocates memory to increase the size of the array.

```
bool Reserve (
    int size      // Number
)
```

### Parameters

*size*

[in] The number of additional elements of the array.

### Return Value

true if successful, false - if there was an attempt to seek the amount is less than or equal to zero, or if the array did not increase.

### Note

To reduce memory fragmentation, increase the size of the array is made with a step previously given through the method of Step (int), or 16 (default).

### Example:

```
//--- example for CArrayShort::Reserve(int)
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
{
    CArrayShort *array=new CArrayShort;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- reserve memory
    if(!array.Reserve(1024))
    {
        printf("Reserve error");
        delete array;
        return;
    }
    //--- use array
    //--- . . .
    //--- delete array
    delete array;
}
```

## Resize

Sets a new (smaller) size of the array.

```
bool Resize(  
    int size      // Size  
)
```

### Parameters

*size*

[in] New size of the array.

### Return Value

true if successful, false - if there was an attempt to set the size of less than or equal to zero.

### Note

Changing the size of the array allows optimal use of memory. Superfluous elements on the right lost. To reduce fragmentation of memory, change the size of the array is made with a step previously given through the method of Step (int), or 16 (default).

### Example:

```
///--- example for CArrayShort::Resize(int)  
#include <Arrays\ArrayShort.mqh>  
///---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    ///--- resize array  
    if(!array.Resize(10))  
    {  
        printf("Resize error");  
        delete array;  
        return;  
    }  
    ///--- delete array  
    delete array;  
}
```

## Shutdown

Clears the array with a full memory release.

```
bool Shutdown()
```

### Return Value

true if successful, false - if an error occurred.

### Example:

```
//--- example for CArrayShort::Shutdown()
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
{
    CArrayShort *array=new CArrayShort;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- add arrays elements
    //--- . . .
    //--- shutdown array
    if(!array.Shutdown())
    {
        printf("Shutdown error");
        delete array;
        return;
    }
    //--- delete array
    delete array;
}
```

## Add

Adds an element to the end of the array.

```
bool Add(  
    short element    // Element to add  
)
```

### Parameters

*element*

[in] Value of the element to add to the array.

### Return Value

true if successful, false - if you can not add an element.

### Example:

```
///--- example for CArrayShort::Add(short)  
#include <Arrays\ArrayShort.mqh>  
///---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    for(int i=0;i<100;i++)  
    {  
        if(!array.Add(i))  
        {  
            printf("Element addition error");  
            delete array;  
            return;  
        }  
    }  
    ///--- use array  
    ///--- . . .  
    ///--- delete array  
    delete array;  
}
```



## AddArray

Adds to the end of the array elements from another array.

```
bool AddArray(  
    const short& src[]      // Source array  
)
```

### Parameters

*src[]*

[in] Reference to an array of source elements to add.

### Return Value

true if successful, false - if you can not add items.

### Example:

```
///--- example for CArrayShort::AddArray(const short &[])  
#include <Arrays\ArrayShort.mqh>  
///---  
short src[];  
///---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add another array  
    if(!array.AddArray(src))  
    {  
        printf("Array addition error");  
        delete array;  
        return;  
    }  
    ///--- use array  
    ///--- . . .  
    ///--- delete array  
    delete array;  
}
```

## AddArray

Adds to the end of the array elements from another array.

```
bool AddArray(  
    const CArrayShort* src      // Pointer to the source  
)
```

### Parameters

*src*

[in] Pointer to an instance of class CArrayShort-source elements to add.

### Return Value

true if successful, false - if you can not add items.

### Example:

```
///--- example for CArrayShort::AddArray(const CArrayShort*)  
#include <Arrays\ArrayShort.mqh>  
///---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- create source array  
    CArrayShort *src=new CArrayShort;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    ///--- add source arrays elements  
    ///--- . . .  
    ///--- add another array  
    if(!array.AddArray(src))  
    {  
        printf("Array addition error");  
        delete src;  
        delete array;  
        return;  
    }  
    ///--- delete source array  
    delete src;
```

```
//--- use array  
//--- . . .  
//--- delete array  
delete array;  
}
```

## Insert

Inserts an element in the array to the specified position.

```
bool Insert(  
    short element,    // Element to insert  
    int pos           // Position  
)
```

### Parameters

*element*

[in] Value of the element to be inserted into an array

*pos*

[in] Position in the array to insert

### Return Value

true if successful, false - if you can not insert the element.

### Example:

```
//--- example for CArrayShort::Insert(short,int)  
#include <Arrays\ArrayShort.mqh>  
//---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- insert elements  
    for(int i=0;i<100;i++)  
    {  
        if(!array.Insert(i,0))  
        {  
            printf("Insert error");  
            delete array;  
            return;  
        }  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

## InsertArray

Inserts an array of elements from another array specified position.

```
bool  InsertArray(  
    const short&  src[],      // Source array  
    int           pos         // Position  
)
```

### Parameters

*src[]*

[in] Reference to an array of source elements to insert

*pos*

[in] Position in the array to insert

### Return Value

true if successful, false - if you can not paste items.

### Example:

```
//--- example for CArrayShort::InsertArray(const short &[],int)  
#include <Arrays\ArrayShort.mqh>  
//---  
short src[];  
//---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- insert another array  
    if(!array.InsertArray(src,0))  
    {  
        printf("Array inserting error");  
        delete array;  
        return;  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

## InsertArray

Inserts an array of elements from another array ukazannogy position.

```
bool InsertArray(  
    CArrayShort* src,      // Pointer to the source  
    int pos              // Position  
)
```

### Parameters

*src*

[in] Pointer to an instance of class CArrayShort-source elements to insert.

*pos*

[in] Position in the array to insert

### Return Value

true if successful, false - if you can not paste items.

### Example:

```
//--- example for CArrayShort::InsertArray(const CArrayShort*,int)  
#include <Arrays\ArrayShort.mqh>  
//---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source array  
    CArrayShort *src=new CArrayShort;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    //--- add source arrays elements  
    //--- . . .  
    //--- insert another array  
    if(!array.InsertArray(src,0))  
    {  
        printf("Array inserting error");  
        delete src;  
        delete array;  
    }  
}
```

```
        return;  
    }  
    //-- delete source array  
    delete src;  
    //-- use array  
    //-- . . .  
    //-- delete array  
    delete array;  
}
```

## AssignArray

Copies the array elements from another array.

```
bool AssignArray(  
    const short& src[]    // Source array  
)
```

### Parameters

*src[]*

[in] Reference to an array of source elements to copy.

### Return Value

true if successful, false - if you can not copy the items.

### Example:

```
///--- example for CArrayShort::AssignArray(const short &[])  
#include <Arrays\ArrayShort.mqh>  
///---  
short src[];  
///---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- assign another array  
    if(!array.AssignArray(src))  
    {  
        printf("Array assigned error");  
        delete array;  
        return;  
    }  
    ///--- use array  
    ///--- . . .  
    ///--- delete array  
    delete array;  
}
```



## AssignArray

Copies the array elements from another array.

```
bool AssignArray(  
    const CArrayShort* src      // Pointer to the source  
)
```

### Parameters

*src*

[in] Pointer to an instance of class CArrayShort-source element to copy.

### Return Value

true if successful, false - if you can not copy the items.

### Example:

```
//--- example for CArrayShort::AssignArray(const CArrayShort*)  
#include <Arrays\ArrayShort.mqh>  
//---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source array  
    CArrayShort *src =new CArrayShort;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    //--- add source arrays elements  
    //--- . . .  
    //--- assign another array  
    if(!array.AssignArray(src))  
    {  
        printf("Array assigned error");  
        delete src;  
        delete array;  
        return;  
    }  
    //--- arrays is identical  
    //--- delete source array
```

```
delete src;  
//--- use array  
//--- . . .  
//--- delete array  
delete array;  
}
```

## Update

Changes the element at the specified position array.

```
bool Update(  
    int    pos,          // Position  
    short  element       // Value  
)
```

### Parameters

*pos*

[in] Position of the element in the array to change

*element*

[in] New value element

### Return Value

true if successful, false - if you can not change the element.

### Example:

```
//--- example for CArrayShort::Update(int,short)  
#include <Arrays\ArrayShort.mqh>  
//---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- update element  
    if(!array.Update(0,100))  
    {  
        printf("Update error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

## Shift

Moves an item from a given position in the array to the specified offset.

```
bool Shift(  
    int pos,          // Positions  
    int shift         // Shift  
)
```

### Parameters

*pos*

[in] Position of the moved element array

*shift*

[in] The value of displacement (both positive and negative).

### Return Value

true if successful, false - if you can not move the item.

### Example:

```
//--- example for CArrayShort::Shift(int,int)  
#include <Arrays\ArrayShort.mqh>  
//---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- shift element  
    if(!array.Shift(10,-5))  
    {  
        printf("Shift error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

## Delete

Removes the element from the given position in the array.

```
bool Delete(  
    int pos    // Position  
)
```

### Parameters

*pos*

[in] Position removes the element in the array.

### Return Value

true if successful, false - if you can not remove the element.

### Example:

```
//--- example for CArrayShort::Delete(int)  
#include <Arrays\ArrayShort.mqh>  
//---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- delete element  
    if(!array.Delete(0))  
    {  
        printf("Delete error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

## DeleteRange

Deletes a group of elements from a given position in the array.

```
bool DeleteRange(  
    int from,      // Position of the first element  
    int to         // Position of the last element  
)
```

### Parameters

*from*

[in] Position of the first removed element in the array.

*to*

[in] Position of the last deleted element in the array.

### Return Value

true if successful, false - if you can not remove elements.

### Example:

```
//--- example for CArrayShort::DeleteRange(int,int)  
#include <Arrays\ArrayShort.mqh>  
//---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- delete elements  
    if(!array.DeleteRange(0,10))  
    {  
        printf("Delete error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

## At

Gets the element from the given position in the array.

```
short At(  
    int pos    // Position  
) const
```

### Parameters

*pos*

[in] Position of the desired element in the array.

### Return Value

The value of the element in case of success, `SHORT_MAX`-if there was an attempt to get an element of not existing positions (the last error `ERR_OUT_OF_RANGE`).

### Note

Of course, `SHORT_MAX` may be a valid value of an array element, so having a value, always check the last error code.

### Example:

```
///--- example for CArrayShort::At(int)  
#include <Arrays\ArrayShort.mqh>  
///---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    for(int i=0;i<array.Total();i++)  
    {  
        short result=array.At(i);  
        if(result==SHORT_MAX && GetLastError()==ERR_OUT_OF_RANGE)  
        {  
            ///--- error of reading from array  
            printf("Get element error");  
            delete array;  
            return;  
        }  
        ///--- use element  
        ///--- . . .  
    }  
}
```

```
//--- delete array  
delete array;  
}
```



## CompareArray

Compares array with another array.

```
bool CompareArray(  
    const short& src[]      // Source array  
    ) const
```

### Parameters

*src[]*

[in] Reference to an array of source elements for comparison.

### Return Value

true if arrays are equal, false - if not.

### Example:

```
//--- example for CArrayShort::CompareArray(const short &[])  
#include <Arrays\ArrayShort.mqh>  
//---  
short src[];  
//---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- compare with another array  
    int result=array.CompareArray(src);  
    //--- delete array  
    delete array;  
}
```

## CompareArray

Compares array with another array.

```
bool CompareArray(  
    const CArrayShort* src      // Pointer to the source  
    ) const
```

### Parameters

*src*

[in] Pointer to an instance of class CArrayShort-source elements for comparison.

### Return Value

true if arrays are equal, false - if not.

### Example:

```
//--- example for CArrayShort::CompareArray(const CArrayShort*)  
#include <Arrays\ArrayShort.mqh>  
//---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source array  
    CArrayShort *src=new CArrayShort;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    //--- add source arrays elements  
    //--- . . .  
    //--- compare with another array  
    int result=array.CompareArray(src);  
    //--- delete arrays  
    delete src;  
    delete array;  
}
```

## InsertSort

Inserts element in a sorted array.

```
bool InsertSort(  
    short element    // Element to insert  
)
```

### Parameters

*element*

[in] Value of the element to be inserted into a sorted array

### Return Value

true if successful, false - if you can not insert the element.

### Example:

```
//--- example for CArrayShort::InsertSort(short)  
#include <Arrays\ArrayShort.mqh>  
//---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- insert element  
    if(!array.InsertSort(100))  
    {  
        printf("Insert error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

## Search

Searches for an element equal to the sample in the sorted array.

```
int Search(  
    short element    // Sample  
    ) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayShort::Search(short)  
#include <Arrays\ArrayShort.mqh>  
//---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.Search(100)!=-1) printf("Element found");  
    else                      printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchGreat

Searches for an element of more samples in sorted array.

```
int SearchGreat(  
    short element    // Sample  
    ) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayShort::SearchGreat(short)  
#include <Arrays\ArrayShort.mqh>  
//---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchGreat(100)!=-1) printf("Element found");  
    else                          printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchLess

Searches for an element less than the sample in the sorted array.

```
int SearchLess(  
    short element    // Sample  
    ) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayShort::SearchLess(short)  
#include <Arrays\ArrayShort.mqh>  
//---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchLess(100)!=-1) printf("Element found");  
    else                          printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchGreatOrEqual

Searches for an element greater than or equal to the sample in the sorted array.

```
int SearchGreatOrEqual(  
    short element    // Sample  
) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayShort::SearchGreatOrEqual(short)  
#include <Arrays\ArrayShort.mqh>  
//---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchGreatOrEqual(100) != -1) printf("Element found");  
    else                                  printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchLessOrEqual

Searches for an element less than or equal to the sample in the sorted array.

```
int SearchLessOrEqual(  
    short element    // Sample  
) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayShort::SearchLessOrEqual(short)  
#include <Arrays\ArrayShort.mqh>  
//---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchLessOrEqual(100) != -1) printf("Element found");  
    else                                printf("Element not found");  
    //--- delete array  
    delete array;  
}
```



## SearchFirst

Finds the first element equal to the sample in the sorted array.

```
int SearchFirst(  
    short element    // Sample  
    ) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayShort::SearchFirst(short)  
#include <Arrays\ArrayShort.mqh>  
//---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchFirst(100)!=-1) printf("Element found");  
    else                          printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchLast

Finds the last element equal to the model in sorted array.

```
int SearchLast(  
    short element    // Sample  
) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayShort::SearchLast(short)  
#include <Arrays\ArrayShort.mqh>  
//---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchLast(100)!=-1) printf("Element found");  
    else                          printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchLinear

Searches for the element equal to the sample in the array.

```
int SearchLinear(  
    short element    // Sample  
) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the element was not found.

### Note

The method uses the linear search (or sequential search) algorithm for unsorted arrays.

### Example:

```
///--- example for CArrayShort::SearchLinear(short)  
#include <Arrays\ArrayShort.mqh>  
///---  
void OnStart()  
{  
    CArrayShort *array=new CArrayShort;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    ///--- search element  
    if(array.SearchLinear(100) != -1) printf("Element found");  
    else                             printf("Element not found");  
    ///--- delete array  
    delete array;  
}
```

## Save

Saves data array in the file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] Handle to open earlier, with the function FileOpen (...), binary file.

### Return Value

true - if successfully completed, false - if an error.

### Example:

```
//--- example for CArrayShort::Save(int)  
#include <Arrays\ArrayShort.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array!=NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add 100 arrays elements  
    for(int i=0;i<100;i++)  
    {  
        array.Add(i);  
    }  
    //--- open file  
    file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!array.Save(file_handle))  
        {  
            //--- file save error  
            printf("File save: Error %d!",GetLastError());  
            delete array;  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }
```

```
    }  
    delete array;  
}
```

## Load

Loads data array from a file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] Handle to open earlier, with the function FileOpen (...), binary file.

### Return Value

true - if successfully completed, false - if an error.

### Example:

```
//--- example for CArrayShort::Load(int)  
#include <Arrays\ArrayShort.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CArrayShort *array=new CArrayShort;  
    //---  
    if(array!=NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- open file  
    file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!array.Load(file_handle))  
        {  
            //--- file load error  
            printf("File load: Error %d!",GetLastError());  
            delete array;  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }  
    //--- use arrays elements  
    for(int i=0;i<array.Total();i++)  
    {  
        printf("Element[%d] = %d",i,array.At(i));  
    }  
}
```

```
    }  
    delete array;  
}
```

## Type

Gets the type identifier of the array.

```
virtual int Type() const
```

### Return Value

ID type of the array (for CArrayShort - 82).

### Example:

```
//--- example for CArrayShort::Type()
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
{
    CArrayShort *array=new CArrayShort;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- get array type
    int type=array.Type();
    //--- delete array
    delete array;
}
```



## CArrayInt

CArrayInt class is a class of dynamic array of variables of type int or uint.

### Description

Class CArrayInt provides the ability to work with a dynamic array of variables of type int or uint. In the class implemented the ability to add / insert / delete elements in an array, sort array, searching in sorted array. In addition, the implemented methods of work with the file.

### Declaration

```
class CArrayInt : public CArray
```

### Title

```
#include <Arrays\ArrayInt.mqh>
```

### Class Methods

| Memory control              |                                                                             |
|-----------------------------|-----------------------------------------------------------------------------|
| <a href="#">Reserve</a>     | Allocates memory to increase the size of the array                          |
| <a href="#">Resize</a>      | Sets a new (smaller) size of the array                                      |
| <a href="#">Shutdown</a>    | Clears the array with a full memory release                                 |
| Add methods                 |                                                                             |
| <a href="#">Add</a>         | Adds an element to the end of the array                                     |
| <a href="#">AddArray</a>    | Adds to the end of the array elements from another array                    |
| <a href="#">AddArray</a>    | Adds to the end of the array elements from another array                    |
| <a href="#">Insert</a>      | Inserts an element in the array to the specified position                   |
| <a href="#">InsertArray</a> | Inserts an array of elements from another array with the specified position |
| <a href="#">InsertArray</a> | Inserts an array of elements from another array with the specified position |
| <a href="#">AssignArray</a> | Copies the array elements from another array                                |
| <a href="#">AssignArray</a> | Copies the array elements from another array                                |
| Update methods              |                                                                             |
| <a href="#">Update</a>      | Changes the element at the specified position array                         |

|                                    |                                                                            |
|------------------------------------|----------------------------------------------------------------------------|
| <a href="#">Shift</a>              | Moves an item from a given position in the array to the specified offset   |
| <b>Delete methods</b>              |                                                                            |
| <a href="#">Delete</a>             | Removes the element from the specified position array                      |
| <a href="#">DeleteRange</a>        | Deletes a group of elements from the specified position array              |
| <b>Access methods</b>              |                                                                            |
| <a href="#">At</a>                 | Gets the element from the specified position array                         |
| <b>Compare methods</b>             |                                                                            |
| <a href="#">CompareArray</a>       | Compares array with another array                                          |
| <a href="#">CompareArray</a>       | Compares array with another array                                          |
| <b>Sorted array operations</b>     |                                                                            |
| <a href="#">InsertSort</a>         | Inserts element in a sorted array                                          |
| <a href="#">Search</a>             | Searches for an element equal to the model in sorted array                 |
| <a href="#">SearchGreat</a>        | Searches for an element of more samples in sorted array                    |
| <a href="#">SearchLess</a>         | Searches for an element less than the sample in the sorted array           |
| <a href="#">SearchGreatOrEqual</a> | Searches for an element greater than or equal to the model in sorted array |
| <a href="#">SearchLessOrEqual</a>  | Searches for an element less than or equal to the model in sorted array    |
| <a href="#">SearchFirst</a>        | Finds the first element equal to the model in sorted array                 |
| <a href="#">SearchLast</a>         | Finds the last element equal to the model in sorted array                  |
| <a href="#">SearchLinear</a>       | Searches for the element equal to the sample in the array                  |
| <b>Input/output</b>                |                                                                            |
| virtual <a href="#">Save</a>       | Saves data array in the file                                               |
| virtual <a href="#">Load</a>       | Loads data array from a file                                               |
| virtual <a href="#">Type</a>       | Gets the type identifier of the array                                      |

## Reserve

Allocates memory to increase the size of the array.

```
bool Reserve (  
    int size      // Number  
)
```

### Parameters

*size*

[in] The number of additional elements of the array.

### Return Value

true if successful, false - if there was an attempt to seek the amount is less than or equal to zero, or if the array did not increase.

### Note

To reduce memory fragmentation, increase the size of the array is made with a step previously given through the method of Step (int), or 16 (default).

### Example:

```
///--- example for CArrayInt::Reserve(int)  
#include <Arrays\ArrayInt.mqh>  
///---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- reserve memory  
    if(!array.Reserve(1024))  
    {  
        printf("Reserve error");  
        delete array;  
        return;  
    }  
    ///--- use array  
    ///--- . . .  
    ///--- delete array  
    delete array;  
}
```

## Resize

Sets a new (smaller) size of the array.

```
bool Resize(  
    int size      // Number  
)
```

### Parameters

*size*

[in] New size of the array.

### Return Value

true if successful, false - if there was an attempt to set the size of less than or equal to zero.

### Note

Changing the size of the array allows optimal use of memory. Superfluous elements on the right lost. To reduce fragmentation of memory, change the size of the array is made with a step previously given through the method of Step (int), or 16 (default).

### Example:

```
//--- example for CArrayInt::Resize(int)  
#include <Arrays\ArrayInt.mqh>  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- resize array  
    if(!array.Resize(10))  
    {  
        printf("Resize error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

## Shutdown

Clears the array with a full memory release.

```
bool Shutdown()
```

### Return Value

true if successful, false - if an error occurred.

### Example:

```
//--- example for CArrayInt::Shutdown()
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
{
    CArrayInt *array=new CArrayInt;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- add arrays elements
    //--- . . .
    //--- shutdown array
    if(!array.Shutdown())
    {
        printf("Shutdown error");
        delete array;
        return;
    }
    //--- delete array
    delete array;
}
```

## Add

Adds an element to the end of the array.

```
bool Add(  
    int element    // Element to add  
)
```

### Parameters

*element*

[in] value of the element to add to the array.

### Return Value

true if successful, false - if you can not add an element.

### Example:

```
///--- example for CArrayInt::Add(int)  
#include <Arrays\ArrayInt.mqh>  
///---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    for(int i=0;i<100;i++)  
    {  
        if(!array.Add(i))  
        {  
            printf("Element addition error");  
            delete array;  
            return;  
        }  
    }  
    ///--- use array  
    ///--- . . .  
    ///--- delete array  
    delete array;  
}
```

## AddArray

Adds to the end of the array elements from another array.

```
bool AddArray(  
    const int& src[]    // Source array  
)
```

### Parameters

*src[]*

[in] Reference to an array of source elements to add.

### Return Value

true if successful, false - if you can not add items.

### Example:

```
///--- example for CArrayInt::AddArray(const int &[])  
#include <Arrays\ArrayInt.mqh>  
///---  
int src[];  
///---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add another array  
    if(!array.AddArray(src))  
    {  
        printf("Array addition error");  
        delete array;  
        return;  
    }  
    ///--- use array  
    ///--- . . .  
    ///--- delete array  
    delete array;  
}
```

## AddArray

Adds to the end of the array elements from another array.

```
bool AddArray(  
    const CArrayInt* src      // Pointer to the source  
)
```

### Parameters

*src*

[in] Pointer to an instance of class CArrayInt-source elements to add.

### Return Value

true if successful, false - if you can not add items.

### Example:

```
//--- example for CArrayInt::AddArray(const CArrayInt*)  
#include <Arrays\ArrayInt.mqh>  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source array  
    CArrayInt *src=new CArrayInt;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    //--- add source arrays elements  
    //--- . . .  
    //--- add another array  
    if(!array.AddArray(src))  
    {  
        printf("Array addition error");  
        delete src;  
        delete array;  
        return;  
    }  
    //--- delete source array  
    delete src;
```



```
//--- use array  
//--- . . .  
//--- delete array  
delete array;  
}
```

## Insert

Inserts an element in the array to the specified position.

```
bool Insert(  
    int element,    // Element to insert  
    int pos         // Position  
)
```

### Parameters

*element*

[in] Value of the element to be inserted into an array

*pos*

[in] Position in the array to insert

### Return Value

true if successful, false - if you can not insert the element.

### Example:

```
//--- example for CArrayInt::Insert(int,int)  
#include <Arrays\ArrayInt.mqh>  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- insert elements  
    for(int i=0;i<100;i++)  
    {  
        if(!array.Insert(i,0))  
        {  
            printf("Insert error");  
            delete array;  
            return;  
        }  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

## InsertArray

Inserts an array of elements from another array specified position.

```
bool  InsertArray(  
    const int&  src[],      // Source array  
    int         pos        // Position  
)
```

### Parameters

*src[]*

[in] Reference to an array of source elements to insert

*pos*

[in] Position in the array to insert

### Return Value

true if successful, false - if you can not paste items.

### Example:

```
//--- example for CArrayInt::InsertArray(const int &[],int)  
#include <Arrays\ArrayInt.mqh>  
//---  
int src[];  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- insert another array  
    if(!array.InsertArray(src,0))  
    {  
        printf("Array inserting error");  
        delete array;  
        return;  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

## InsertArray

Inserts an array of elements from another array specified position.

```
bool InsertArray(  
    CArrayInt* src,      // Pointer to the source  
    int pos             // Position  
)
```

### Parameters

*src*

[in] Pointer to an instance of class CArrayInt-source elements to insert.

*pos*

[in] Position in the array to insert.

### Return Value

true if successful, false - if you can not paste items.

### Example:

```
//--- example for CArrayInt::InsertArray(const CArrayInt*,int)  
#include <Arrays\ArrayInt.mqh>  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source array  
    CArrayInt *src=new CArrayInt;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    //--- add source arrays elements  
    //--- . . .  
    //--- insert another array  
    if(!array.InsertArray(src,0))  
    {  
        printf("Array inserting error");  
        delete src;  
        delete array;  
    }  
}
```

```
        return;  
    }  
    //--- delete source array  
    delete src;  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

## AssignArray

Copies the array elements from another array.

```
bool AssignArray(  
    const int& src[]    // Source array  
)
```

### Parameters

*src[]*

[in] Reference to an array of source elements to copy.

### Return Value

true if successful, false - if you can not copy the items.

### Example:

```
///--- example for CArrayInt::AssignArray(const int &[])  
#include <Arrays\ArrayInt.mqh>  
///---  
int src[];  
///---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- assign another array  
    if(!array.AssignArray(src))  
    {  
        printf("Array assigned error");  
        delete array;  
        return;  
    }  
    ///--- use array  
    ///--- . . .  
    ///--- delete array  
    delete array;  
}
```

## AssignArray

Copies the array elements from another array.

```
bool AssignArray(  
    const CArrayInt* src      // Pointer to the source  
)
```

### Parameters

*src*

[in] Pointer to an instance of class CArrayInt-source element to copy.

### Return Value

true if successful, false - if you can not copy the items.

### Example:

```
//--- example for CArrayInt::AssignArray(const CArrayInt*)  
#include <Arrays\ArrayInt.mqh>  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source array  
    CArrayInt *src =new CArrayInt;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    //--- add source arrays elements  
    //--- . . .  
    //--- assign another array  
    if(!array.AssignArray(src))  
    {  
        printf("Array assigned error");  
        delete src;  
        delete array;  
        return;  
    }  
}
```

```
//--- arrays is identical
//--- delete source array
delete src;
//--- use array
//--- . . .
//--- delete array
delete array;
}
```



## Update

Changes the element at the specified position array.

```
bool Update(  
    int pos,          // Position  
    int element       // Value  
)
```

### Parameters

*pos*

[in] Position of the element in the array to change.

*element*

[in] New value element

### Return Value

true if successful, false - if you can not change the element.

### Example:

```
//--- example for CArrayInt::Update(int,int)  
#include <Arrays\ArrayInt.mqh>  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- update element  
    if(!array.Update(0,10000))  
    {  
        printf("Update error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

## Shift

Moves an item from a given position in the array to the specified offset.

```
bool Shift(  
    int pos,          // Position  
    int shift         // Shift  
)
```

### Parameters

*pos*

[in] Position of the moved element array

*shift*

[in] The value of displacement (both positive and negative).

### Return Value

true if successful, false - if you can not move the item.

### Example:

```
//--- example for CArrayInt::Shift(int,int)  
#include <Arrays\ArrayInt.mqh>  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- shift element  
    if(!array.Shift(10,-5))  
    {  
        printf("Shift error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

## Delete

Removes the element from the given position in the array.

```
bool Delete(  
    int pos    // Position  
)
```

### Parameters

*pos*

[in] Position removes the element in the array.

### Return Value

true if successful, false - if you can not remove the element.

### Example:

```
///--- example for CArrayInt::Delete(int)  
#include <Arrays\ArrayInt.mqh>  
///---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    ///--- delete element  
    if(!array.Delete(0))  
    {  
        printf("Delete error");  
        delete array;  
        return;  
    }  
    ///--- delete array  
    delete array;  
}
```

## DeleteRange

Deletes a group of elements from a given position in the array.

```
bool DeleteRange(  
    int from,      // Position of the first element  
    int to         // Position of the last element  
)
```

### Parameters

*from*

[in] Position of the first removes the element in the array.

*to*

[in] Position of the last deleted element in the array.

### Return Value

true if successful, false - if you can not remove elements.

### Example:

```
//--- example for CArrayInt::DeleteRange(int,int)  
#include <Arrays\ArrayInt.mqh>  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- delete elements  
    if(!array.DeleteRange(0,10))  
    {  
        printf("Delete error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

## At

Gets the element from the given position in the array.

```
int At(  
    int pos    // Position  
) const
```

### Parameters

*pos*

[in] Position of the desired element in the array.

### Return Value

The value of the element in case of success, INT\_MAX-if there was an attempt to get an element of not existing positions (the last error ERR\_OUT\_OF\_RANGE).

### Note

Of course, INT\_MAX could be a valid value of an array element, so having a value, always check the last error code.

### Example:

```
//--- example for CArrayInt::At(int)  
#include <Arrays\ArrayInt.mqh>  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    for(int i=0;i<array.Total();i++)  
    {  
        int result=array.At(i);  
        if(result==INT_MAX && GetLastError()==ERR_OUT_OF_RANGE)  
        {  
            //--- error of reading from array  
            printf("Get element error");  
            delete array;  
            return;  
        }  
        //--- use element  
        //--- . . .  
    }  
}
```

```
//--- delete array  
delete array;  
}
```

## CompareArray

Compares array with another array.

```
bool CompareArray(  
    const int& src[]    // Source array  
    ) const
```

### Parameters

*src[]*

[in] Reference to an array of source elements for comparison.

### Return Value

true if arrays are equal, false - if not.

### Example:

```
//--- example for CArrayInt::CompareArray(const int &[])  
#include <Arrays\ArrayInt.mqh>  
//---  
int src[];  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- compare with another array  
    int result=array.CompareArray(src);  
    //--- delete array  
    delete array;  
}
```

## CompareArray

Compares array with another array.

```
bool CompareArray(  
    const CArrayInt* src      // Pointer to the source  
    ) const
```

### Parameters

*src*

[in] Pointer to an instance of class CArrayInt-source elements for comparison.

### Return Value

true if arrays are equal, false - if not.

### Example:

```
///--- example for CArrayInt::CompareArray(const CArrayInt*)  
#include <Arrays\ArrayInt.mqh>  
///---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- create source array  
    CArrayInt *src=new CArrayInt;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    ///--- add source arrays elements  
    ///--- . . .  
    ///--- compare with another array  
    int result=array.CompareArray(src);  
    ///--- delete arrays  
    delete src;  
    delete array;  
}
```



## InsertSort

Inserts element in a sorted array.

```
bool InsertSort(  
    int element    // Element to insert  
)
```

### Parameters

*element*

[in] value of the element to be inserted into a sorted array

### Return Value

true if successful, false - if you can not insert the element.

### Example:

```
///--- example for CArrayInt::InsertSort(int)  
#include <Arrays\ArrayInt.mqh>  
///---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    ///--- sort array  
    array.Sort();  
    ///--- insert element  
    if(!array.InsertSort(10000))  
    {  
        printf("Insert error");  
        delete array;  
        return;  
    }  
    ///--- delete array  
    delete array;  
}
```

## Search

Searches for an element equal to the sample in the sorted array.

```
int Search(  
    int element    // Sample  
) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayInt::Search(int)  
#include <Arrays\ArrayInt.mqh>  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.Search(10000) != -1) printf("Element found");  
    else                          printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchGreat

Searches for an element of more samples in sorted array.

```
int SearchGreat(  
    int element    // Sample  
) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayInt::SearchGreat(int)  
#include <Arrays\ArrayInt.mqh>  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchGreat(10000) != -1) printf("Element found");  
    else                             printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchLess

Searches for an element less than the sample in the sorted array.

```
int SearchLess(  
    int element    // Sample  
    ) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayInt::SearchLess(int)  
#include <Arrays\ArrayInt.mqh>  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchLess(10000) != -1) printf("Element found");  
    else                             printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchGreatOrEqual

Searches for an element greater than or equal to the sample in the sorted array.

```
int SearchGreatOrEqual(  
    int element    // Element to search  
) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayInt::SearchGreatOrEqual(int)  
#include <Arrays\ArrayInt.mqh>  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchGreatOrEqual(10000)!=-1) printf("Element found");  
    else                                  printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchLessOrEqual

Searches for an element less than or equal to the sample in the sorted array.

```
int SearchLessOrEqual(  
    int element    // Sample  
) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayInt::SearchLessOrEqual(int)  
#include <Arrays\ArrayInt.mqh>  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchLessOrEqual(10000)!=-1) printf("Element found");  
    else                                printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchFirst

Finds the first element equal to the sample in the sorted array.

```
int SearchFirst(  
    int element    // Sample  
) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayInt:: SearchFirst(int)  
#include <Arrays\ArrayInt.mqh>  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchFirst(10000) != -1) printf("Element found");  
    else                             printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchLast

Finds the last element equal to the model in sorted array.

```
int SearchLast(  
    int element    // Sample  
    ) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayInt::SearchLast(int)  
#include <Arrays\ArrayInt.mqh>  
//---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchLast(10000) != -1) printf("Element found");  
    else                             printf("Element not found");  
    //--- delete array  
    delete array;  
}
```



## SearchLinear

Searches for the element equal to the sample in the array.

```
int SearchLinear(  
    int element    // Sample  
) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the element was not found.

### Note

The method uses the linear search (or sequential search) algorithm for unsorted arrays.

### Example:

```
///--- example for CArrayInt::SearchLinear(int)  
#include <Arrays\ArrayInt.mqh>  
///---  
void OnStart()  
{  
    CArrayInt *array=new CArrayInt;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    ///--- search element  
    if(array.SearchLinear(10000) != -1) printf("Element found");  
    else                               printf("Element not found");  
    ///--- delete array  
    delete array;  
}
```

## Save

Saves data array in the file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] Handle to open earlier, with the function FileOpen (...), binary file.

### Return Value

true - if successfully completed, false - if an error.

### Example:

```
//--- example for CArrayInt::Save(int)  
#include <Arrays\ArrayInt.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array!=NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add 100 arrays elements  
    for(int i=0;i<100;i++)  
    {  
        array.Add(i);  
    }  
    //--- open file  
    file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!array.Save(file_handle))  
        {  
            //--- file save error  
            printf("File save: Error %d!",GetLastError());  
            delete array;  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }
```

```
    }  
    delete array;  
}
```

## Load

Loads data array from a file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] Handle to open earlier, with the function `FileOpen (...)`, binary file.

### Return Value

true - if successfully completed, false - if an error.

### Example:

```
//--- example for CArrayInt::Load(int)  
#include <Arrays\ArrayInt.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CArrayInt *array=new CArrayInt;  
    //---  
    if(array!=NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- open file  
    file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!array.Load(file_handle))  
        {  
            //--- file load error  
            printf("File load: Error %d!",GetLastError());  
            delete array;  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }  
    //--- use arrays elements  
    for(int i=0;i<array.Total();i++)  
    {  
        printf("Element[%d] = %d",i,array.At(i));  
    }  
}
```

```
    }  
    delete array;  
}
```

## Type

Gets the type identifier of the array.

```
virtual int Type() const
```

### Return Value

ID type of the array (for CArrayInt - 82).

### Example:

```
//--- example for CArrayInt::Type()
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
{
    CArrayInt *array=new CArrayInt;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- get array type
    int type=array.Type();
    //--- delete array
    delete array;
}
```

## CArrayLong

CArrayLong class is a class of dynamic array of variables of type long or ulong.

### Description

Class CArrayLong provides the ability to work with a dynamic array of variables of type long or ulong. In the class implemented the ability to add / insert / delete elements in an array, sort array, searching in sorted array. In addition, the implemented methods of work with the file.

### Declaration

```
class CArrayLong : public CArray
```

### Title

```
#include <Arrays\ArrayLong.mqh>
```

### Class Methods

| Memory control              |                                                                             |
|-----------------------------|-----------------------------------------------------------------------------|
| <a href="#">Reserve</a>     | Allocates memory to increase the size of the array                          |
| <a href="#">Resize</a>      | Sets a new (smaller) size of the array                                      |
| <a href="#">Shutdown</a>    | Clears the array with a full memory release                                 |
| Add methods                 |                                                                             |
| <a href="#">Add</a>         | Adds an element to the end of the array                                     |
| <a href="#">AddArray</a>    | Adds to the end of the array elements from another array                    |
| <a href="#">AddArray</a>    | Adds to the end of the array elements from another array                    |
| <a href="#">Insert</a>      | Inserts an element in the array to the specified position                   |
| <a href="#">InsertArray</a> | Inserts an array of elements from another array with the specified position |
| <a href="#">InsertArray</a> | Inserts an array of elements from another array with the specified position |
| <a href="#">AssignArray</a> | Copies the array elements from another array                                |
| <a href="#">AssignArray</a> | Copies the array elements from another array                                |
| Update methods              |                                                                             |
| <a href="#">Update</a>      | Changes the element at the specified position array                         |

|                                    |                                                                            |
|------------------------------------|----------------------------------------------------------------------------|
| <a href="#">Shift</a>              | Moves an item from a given position in the array to the specified offset   |
| <b>Delete methods</b>              |                                                                            |
| <a href="#">Delete</a>             | Removes the element from the specified position array                      |
| <a href="#">DeleteRange</a>        | Deletes a group of elements from the specified position array              |
| <b>Access methods</b>              |                                                                            |
| <a href="#">At</a>                 | Gets the element from the specified position array                         |
| <b>Compare methods</b>             |                                                                            |
| <a href="#">CompareArray</a>       | Compares array with another array                                          |
| <a href="#">CompareArray</a>       | Compares array with another array                                          |
| <b>Sorted array operations</b>     |                                                                            |
| <a href="#">InsertSort</a>         | Inserts element in a sorted array                                          |
| <a href="#">Search</a>             | Searches for an element equal to the model in sorted array                 |
| <a href="#">SearchGreat</a>        | Searches for an element of more samples in sorted array                    |
| <a href="#">SearchLess</a>         | Searches for an element less than the sample in the sorted array           |
| <a href="#">SearchGreatOrEqual</a> | Searches for an element greater than or equal to the model in sorted array |
| <a href="#">SearchLessOrEqual</a>  | Searches for an element less than or equal to the model in sorted array    |
| <a href="#">SearchFirst</a>        | Finds the first element equal to the model in sorted array                 |
| <a href="#">SearchLast</a>         | Finds the last element equal to the model in sorted array                  |
| <a href="#">SearchLinear</a>       | Searches for the element equal to the sample in the array                  |
| <b>Input/output</b>                |                                                                            |
| virtual <a href="#">Save</a>       | Saves data array in the file                                               |
| virtual <a href="#">Load</a>       | Loads data array from a file                                               |
| virtual <a href="#">Type</a>       | Gets the type identifier of the array                                      |



## Reserve

Allocates memory to increase the size of the array.

```
bool Reserve (
    int size      // Number
)
```

### Parameters

*size*

[in] The number of additional elements of the array.

### Return Value

true if successful, false - if there was an attempt to seek the amount is less than or equal to zero, or if the array did not increase.

### Note

To reduce memory fragmentation, increase the size of the array is made with a step previously given through the method of Step (int), or 16 (default).

### Example:

```
///--- example for CArrayLong::Reserve(int)
#include <Arrays\ArrayLong.mqh>
///---
void OnStart()
{
    CArrayLong *array=new CArrayLong;
    ///---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    ///--- reserve memory
    if(!array.Reserve(1024))
    {
        printf("Reserve error");
        delete array;
        return;
    }
    ///--- use array
    ///--- . . .
    ///--- delete array
    delete array;
}
```

## Resize

Sets a new (smaller) size of the array.

```
bool Resize(  
    int size      // Size  
)
```

### Parameters

*size*

[in] New size of the array.

### Return Value

true if successful, false - if there was an attempt to set the size of less than or equal to zero.

### Note

Changing the size of the array allows optimal use of memory. Superfluous elements on the right lost. To reduce fragmentation of memory, change the size of the array is made with a step previously given through the method of Step (int), or 16 (default).

### Example:

```
/*--- example for CArrayLong::Resize(int)  
#include <Arrays\ArrayLong.mqh>  
/*---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    /*---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    /*--- add arrays elements  
    /*--- . . .  
    /*--- resize array  
    if(!array.Resize(10))  
    {  
        printf("Resize error");  
        delete array;  
        return;  
    }  
    /*--- delete array  
    delete array;  
}
```

## Shutdown

Clears the array with a full memory release.

```
bool Shutdown()
```

### Return Value

true if successful, false - if an error occurred.

### Example:

```
//--- example for CArrayLong::Shutdown()
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
{
    CArrayLong *array=new CArrayLong;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- add arrays elements
    //--- . . .
    //--- shutdown array
    if(!array.Shutdown())
    {
        printf("Shutdown error");
        delete array;
        return;
    }
    //--- delete array
    delete array;
}
```

## Add

Adds an element to the end of the array.

```
bool Add(  
    long element    // Element to add  
)
```

### Parameters

*element*

[in] Value of the element to add to the array.

### Return Value

true if successful, false - if you can not add an element.

### Example:

```
///--- example for CArrayLong::Add(long)  
#include <Arrays\ArrayLong.mqh>  
///---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    for(int i=0;i<100;i++)  
    {  
        if(!array.Add(i))  
        {  
            printf("Element addition error");  
            delete array;  
            return;  
        }  
    }  
    ///--- use array  
    ///--- . . .  
    ///--- delete array  
    delete array;  
}
```

## AddArray

Adds to the end of the array elements from another array.

```
bool AddArray(  
    const long& src[]      // Source array  
)
```

### Parameters

*src[]*

[in] Reference to an array of source elements to add.

### Return Value

true if successful, false - if you can not add items.

### Example:

```
///--- example for CArrayLong::AddArray(const long &[])  
#include <Arrays\ArrayLong.mqh>  
///---  
long src[];  
///---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add another array  
    if(!array.AddArray(src))  
    {  
        printf("Array addition error");  
        delete array;  
        return;  
    }  
    ///--- use array  
    ///--- . . .  
    ///--- delete array  
    delete array;  
}
```

## AddArray

Adds to the end of the array elements from another array.

```
bool AddArray(  
    const CArrayLong* src      // Pointer to the source  
)
```

### Parameters

*src*

[in] Pointer to an instance of class CArrayLong-source elements to add.

### Return Value

true if successful, false - if you can not add items.

### Example:

```
///--- example for CArrayLong::AddArray(const CArrayLong*)  
#include <Arrays\ArrayLong.mqh>  
///---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- create source array  
    CArrayLong *src=new CArrayLong;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    ///--- add source arrays elements  
    ///--- . . .  
    ///--- add another array  
    if(!array.AddArray(src))  
    {  
        printf("Array addition error");  
        delete src;  
        delete array;  
        return;  
    }  
    ///--- delete source array  
    delete src;
```

```
//--- use array  
//--- . . .  
//--- delete array  
delete array;  
}
```

## Insert

Inserts an element in the array to the specified position.

```
bool Insert(  
    long element,    // Element to insert  
    int pos          // Position  
)
```

### Parameters

*element*

[in] Value of the element to be inserted into an array

*pos*

[in] Position in the array to insert

### Return Value

true if successful, false - if you can not insert the element.

### Example:

```
//--- example for CArrayLong::Insert(long,int)  
#include <Arrays\ArrayLong.mqh>  
//---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- insert elements  
    for(int i=0;i<100;i++)  
    {  
        if(!array.Insert(i,0))  
        {  
            printf("Insert error");  
            delete array;  
            return;  
        }  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```



## InsertArray

Inserts an array of elements from another array ukazannogy position.

```
bool InsertArray(  
    const long& src[],      // Source array  
    int pos                // Position  
)
```

### Parameters

*src[]*

[in] Reference to an array of source elements to insert

*pos*

[in] Position in the array to insert

### Return Value

true if successful, false - if you can not paste items.

### Example:

```
//--- example for CArrayLong::InsertArray(const long &[],int)  
#include <Arrays\ArrayLong.mqh>  
//---  
long src[];  
//---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- insert another array  
    if(!array.InsertArray(src,0))  
    {  
        printf("Array inserting error");  
        delete array;  
        return;  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

## InsertArray

Inserts an array of elements from another array ukazannogy position.

```
bool InsertArray(  
    CArrayLong* src,      // Pointer to the source  
    int pos              // Position  
)
```

### Parameters

*src*

[in] Pointer to an instance of class CArrayLong-source elements to insert.

*pos*

[in] Position in the array to insert

### Return Value

true if successful, false - if you can not paste items.

### Example:

```
//--- example for CArrayLong::InsertArray(const CArrayLong*,int)  
#include <Arrays\ArrayLong.mqh>  
//---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source array  
    CArrayLong *src=new CArrayLong;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    //--- add source arrays elements  
    //--- . . .  
    //--- insert another array  
    if(!array.InsertArray(src,0))  
    {  
        printf("Array inserting error");  
        delete src;  
        delete array;  
    }  
}
```

```
        return;  
    }  
    //-- delete source array  
    delete src;  
    //-- use array  
    //-- . . .  
    //-- delete array  
    delete array;  
}
```

## AssignArray

Copies the array elements from another array.

```
bool AssignArray(  
    const long& src[]      // Source array  
)
```

### Parameters

*src[]*

[in] Reference to an array of source elements to copy.

### Return Value

true if successful, false - if you can not copy the items.

### Example:

```
//--- example for CArrayLong::AssignArray(const long &[])  
#include <Arrays\ArrayLong.mqh>  
//---  
long src[];  
//---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- assign another array  
    if(!array.AssignArray(src))  
    {  
        printf("Array assigned error");  
        delete array;  
        return;  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

## AssignArray

Copies the array elements from another array.

```
bool AssignArray(  
    const CArrayLong* src      // Pointer to the source  
)
```

### Parameters

*src*

[in] Pointer to an instance of class CArrayLong-source element to copy.

### Return Value

true if successful, false - if you can not copy the items.

### Example:

```
///--- example for CArrayLong::AssignArray(const CArrayLong*)  
#include <Arrays\ArrayLong.mqh>  
///---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- create source array  
    CArrayLong *src =new CArrayLong;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    ///--- add source arrays elements  
    ///--- . . .  
    ///--- assign another array  
    if(!array.AssignArray(src))  
    {  
        printf("Array assigned error");  
        delete src;  
        delete array;  
        return;  
    }  
    ///--- arrays is identical  
    ///--- delete source array
```

```
delete src;  
//--- use array  
//--- . . .  
//--- delete array  
delete array;  
}
```

## Update

Changes the element at the specified position array.

```
bool Update(  
    int    pos,           // Position  
    long   element       // Value  
)
```

### Parameters

*pos*

[in] Position of the element in the array to change

*element*

[in] New value element

### Return Value

true if successful, false - if you can not change the element.

### Example:

```
//--- example for CArrayLong::Update(int,long)  
#include <Arrays\ArrayLong.mqh>  
//---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- update element  
    if(!array.Update(0,1000000))  
    {  
        printf("Update error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

## Shift

Moves an item from a given position in the array to the specified offset.

```
bool Shift(  
    int pos,          // Position  
    int shift         // Shift  
)
```

### Parameters

*pos*

[in] Position of the moved element array

*shift*

[in] The value of displacement (both positive and negative).

### Return Value

true if successful, false - if you can not move the item.

### Example:

```
//--- example for CArrayLong::Shift(int,int)  
#include <Arrays\ArrayLong.mqh>  
//---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- shift element  
    if(!array.Shift(10,-5))  
    {  
        printf("Shift error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```



## Delete

Removes the element from the given position in the array.

```
bool Delete(  
    int pos    // Position  
)
```

### Parameters

*pos*

[in] Position removes the element in the array.

### Return Value

true if successful, false - if you can not remove the element.

### Example:

```
///--- example for CArrayLong::Delete(int)  
#include <Arrays\ArrayLong.mqh>  
///---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    ///--- delete element  
    if(!array.Delete(0))  
    {  
        printf("Delete error");  
        delete array;  
        return;  
    }  
    ///--- delete array  
    delete array;  
}
```

## DeleteRange

Deletes a group of elements from a given position in the array.

```
bool DeleteRange(  
    int from,      // Position of the first element  
    int to         // Position of the last element  
)
```

### Parameters

*from*

[in] Position of the first removes the element in the array.

*to*

[in] Position of the last deleted element in the array.

### Return Value

true if successful, false - if you can not remove elements.

### Example:

```
//--- example for CArrayLong::DeleteRange(int,int)  
#include <Arrays\ArrayLong.mqh>  
//---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- delete elements  
    if(!array.DeleteRange(0,10))  
    {  
        printf("Delete error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

## At

Gets the element from the given position in the array.

```
long At(  
    int pos    // Position  
) const
```

### Parameters

*pos*

[in] Position of the desired element in the array.

### Return Value

The value of the element in case of success, LONG\_MAX if there was an attempt to get an element of not existing positions (the last error ERR\_OUT\_OF\_RANGE).

### Note

Of course, LONG\_MAX may be a valid value of an array element, so having a value, always check the last error code.

### Example:

```
//--- example for CArrayLong::At(int)  
#include <Arrays\ArrayLong.mqh>  
//---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    for(int i=0;i<array.Total();i++)  
    {  
        long result=array.At(i);  
        if(result==LONG_MAX && GetLastError()==ERR_OUT_OF_RANGE)  
        {  
            //--- Error reading from the array  
            printf("Get element error");  
            delete array;  
            return;  
        }  
        //--- use element  
        //--- . . .  
    }  
}
```

```
//--- delete array  
delete array;  
}
```

## CompareArray

Compares array with another array.

```
bool CompareArray(  
    const long& src[]      // Source array  
    ) const
```

### Parameters

*src[]*

[in] Reference to an array of source elements for comparison.

### Return Value

true if arrays are equal, false - if not.

### Example:

```
//--- example for CArrayLong::CompareArray(const long &[])  
#include <Arrays\ArrayLong.mqh>  
//---  
long src[];  
//---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- compare with another array  
    int result=array.CompareArray(src);  
    //--- delete array  
    delete array;  
}
```

## CompareArrayconst

Compares array with another array.

```
bool CompareArrayconst(  
    const CArrayLong* src      // Pointer to the source  
    ) const
```

### Parameters

*src*

[in] Pointer to an instance of class CArrayLong-source elements for comparison.

### Return Value

true if arrays are equal, false - if not.

### Example:

```
//--- example for CArrayLong::CompareArray(const CArrayLong*)  
#include <Arrays\ArrayLong.mqh>  
//---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source array  
    CArrayLong *src=new CArrayLong;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    //--- add source arrays elements  
    //--- . . .  
    //--- compare with another array  
    int result=array.CompareArray(src);  
    //--- delete arrays  
    delete src;  
    delete array;  
}
```

## InsertSort

Inserts element in a sorted array.

```
bool InsertSort(  
    long element    // Element to insert  
)
```

### Parameters

*element*

[in] Value of the element to be inserted into a sorted array

### Return Value

true if successful, false - if you can not insert the element.

### Example:

```
///--- example for CArrayLong::InsertSort(long)  
#include <Arrays\ArrayLong.mqh>  
///---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    ///--- sort array  
    array.Sort();  
    ///--- insert element  
    if(!array.InsertSort(1000000))  
    {  
        printf("Insert error");  
        delete array;  
        return;  
    }  
    ///--- delete array  
    delete array;  
}
```

## Search

Searches for an element equal to the model in sorted array.

```
int Search(  
    long element    // Sample  
    ) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayLong::Search(long)  
#include <Arrays\ArrayLong.mqh>  
//---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.Search(1000000)!=-1) printf("Element found");  
    else                          printf("Element not found");  
    //--- delete array  
    delete array;  
}
```



## SearchGreat

Searches for an element of larger sample, in sorted array.

```
int SearchGreat(  
    long element    // Sample  
    ) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayLong::SearchGreat(long)  
#include <Arrays\ArrayLong.mqh>  
//---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchGreat(1000000)!=-1) printf("Element found");  
    else                             printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchLess

Searches for an element less than the sample in the sorted array.

```
int SearchLess(  
    long element    // Sample  
    ) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayLong::SearchLess(long)  
#include <Arrays\ArrayLong.mqh>  
//---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchLess(1000000)!=-1) printf("Element found");  
    else                             printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchGreatOrEqual

Searches for an element greater than or equal to the model in sorted array.

```
int SearchGreatOrEqual(  
    long element    // Sample  
) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayLong::SearchGreatOrEqual(long)  
#include <Arrays\ArrayLong.mqh>  
//---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchGreatOrEqual(1000000)!=-1) printf("Element found");  
    else                                     printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchLessOrEqual

Searches for an element less than or equal to the model in sorted array.

```
int SearchLessOrEqual(  
    long element    // Sample  
) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayLong::SearchLessOrEqual(long)  
#include <Arrays\ArrayLong.mqh>  
//---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchLessOrEqual(1000000) != -1) printf("Element found");  
    else                                     printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchFirst

Finds the first element equal to the model in sorted array.

```
int SearchFirst(  
    long element    // Sample  
    ) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayLong::SearchFirst(long)  
#include <Arrays\ArrayLong.mqh>  
//---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchFirst(1000000)!=-1) printf("Element found");  
    else                             printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchLast

Finds the last element equal to the model in sorted array.

```
int SearchLast(  
    long element    // Sample  
    ) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayLong::SearchLast(long)  
#include <Arrays\ArrayLong.mqh>  
//---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchLast(1000000)!=-1) printf("Element found");  
    else                             printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchLinear

Searches for the element equal to the model in the array.

```
int SearchLinear(  
    long element    // Sample  
    ) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the element was not found.

### Note

The method uses the linear search (or sequential search) algorithm for unsorted arrays.

### Example:

```
///--- example for CArrayLong::SearchLinear(long)  
#include <Arrays\ArrayLong.mqh>  
///---  
void OnStart()  
{  
    CArrayLong *array=new CArrayLong;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    ///--- search element  
    if(array.SearchLinear(1000000) != -1) printf("Element found");  
    else                                printf("Element not found");  
    ///--- delete array  
    delete array;  
}
```

## Save

Saves data array in the file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] Handle to open earlier, with the function `FileOpen (...)`, binary file.

### Return Value

true - if successfully completed, false - if an error.

### Example:

```
//--- example for CArrayLong::Save(int)  
#include <Arrays\ArrayLong.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array!=NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add 100 arrays elements  
    for(int i=0;i<100;i++)  
    {  
        array.Add(i);  
    }  
    //--- open file  
    file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!array.Save(file_handle))  
        {  
            //--- file save error  
            printf("File save: Error %d!",GetLastError());  
            delete array;  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }
```



```
    }  
    delete array;  
}
```

## Load

Loads data array from a file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] Handle to open earlier, with the function `FileOpen (...)`, binary file.

### Return Value

true - if successfully completed, false - if an error.

### Example:

```
//--- example for CArrayLong::Load(int)  
#include <Arrays\ArrayLong.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CArrayLong *array=new CArrayLong;  
    //---  
    if(array!=NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- open file  
    file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!array.Load(file_handle))  
        {  
            //--- file load error  
            printf("File load: Error %d!",GetLastError());  
            delete array;  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }  
    //--- use arrays elements  
    for(int i=0;i<array.Total();i++)  
    {  
        printf("Element[%d] = %I64",i,array.At(i));  
    }  
}
```

```
    }  
    delete array;  
}
```

## Type

Gets the type identifier of the array.

```
virtual int Type() const
```

### Return Value

ID type of the array (for CArrayLong - 84).

### Example:

```
//--- example for CArrayLong::Type()
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
{
    CArrayLong *array=new CArrayLong;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- get array type
    int type=array.Type();
    //--- delete array
    delete array;
}
```

## CArrayFloat

CArrayFloat class is a class of dynamic array of variables of type float.

### Description

Class CArrayFloat provides the ability to work with a dynamic array of variables of type float. In the class implemented the ability to add / insert / delete elements in an array, sort array, searching in sorted array. In addition, the implemented methods of work with the file.

### Declaration

```
class CArrayFloat : public CArray
```

### Title

```
#include <Arrays\ArrayFloat.mqh>
```

### Class Methods

| Attributes                  |                                                                             |
|-----------------------------|-----------------------------------------------------------------------------|
| <a href="#">Delta</a>       | Set the comparison tolerance                                                |
| Memory control              |                                                                             |
| <a href="#">Reserve</a>     | Allocates memory to increase the size of the array                          |
| <a href="#">Resize</a>      | Sets a new (smaller) size of the array                                      |
| <a href="#">Shutdown</a>    | Clears the array with a full memory release                                 |
| Add methods                 |                                                                             |
| <a href="#">Add</a>         | Adds an element to the end of the array                                     |
| <a href="#">AddArray</a>    | Adds to the end of the array elements from another array                    |
| <a href="#">AddArray</a>    | Adds to the end of the array elements from another array                    |
| <a href="#">Insert</a>      | Inserts an element in the array to the specified position                   |
| <a href="#">InsertArray</a> | Inserts an array of elements from another array with the specified position |
| <a href="#">InsertArray</a> | Inserts an array of elements from another array with the specified position |
| <a href="#">AssignArray</a> | Copies the array elements from another array                                |
| <a href="#">AssignArray</a> | Copies the array elements from another array                                |
| Update methods              |                                                                             |

|                                    |                                                                            |
|------------------------------------|----------------------------------------------------------------------------|
| <a href="#">Update</a>             | Changes the element at the specified position array                        |
| <a href="#">Shift</a>              | Moves an item from a given position in the array to the specified offset   |
| <a href="#">Delete</a>             | Removes the element from the specified position array                      |
| <a href="#">DeleteRange</a>        | Deletes a group of elements from the specified position array              |
| <b>Access methods</b>              |                                                                            |
| <a href="#">At</a>                 | Gets the element from the specified position array                         |
| <b>Compare methods</b>             |                                                                            |
| <a href="#">CompareArray</a>       | Compares array with another array                                          |
| <a href="#">CompareArray</a>       | Compares array with another array                                          |
| <b>Sorted array operations</b>     |                                                                            |
| <a href="#">InsertSort</a>         | Inserts element in a sorted array                                          |
| <a href="#">Search</a>             | Searches for an element equal to the model in sorted array                 |
| <a href="#">SearchGreat</a>        | Searches for an element of more samples in sorted array                    |
| <a href="#">SearchLess</a>         | Searches for an element less than the sample in the sorted array           |
| <a href="#">SearchGreatOrEqual</a> | Searches for an element greater than or equal to the model in sorted array |
| <a href="#">SearchLessOrEqual</a>  | Searches for an element less than or equal to the model in sorted array    |
| <a href="#">SearchFirst</a>        | Finds the first element equal to the model in sorted array                 |
| <a href="#">SearchLast</a>         | Finds the last element equal to the model in sorted array                  |
| <a href="#">SearchLinear</a>       | Searches for the element equal to the sample in the array                  |
| <b>Input/output</b>                |                                                                            |
| virtual <a href="#">Save</a>       | Saves data array in the file                                               |
| virtual <a href="#">Load</a>       | Loads data array from a file                                               |
| virtual <a href="#">Type</a>       | Gets the type identifier of the array                                      |

## Delta

Set tolerance comparison.

```
void Delta(  
    float delta    // Tolerance  
)
```

### Parameters

*delta*

[in] the new value of the admission of comparison.

### Return Value

None

### Note

Admission of comparison used in the search. Values are considered equal if their difference is less than or equal to tolerance. The default tolerance is 0.0.

### Example:

```
//--- example for CArrayFloat::Delta(float)  
#include <Arrays\ArrayFloat.mqh>  
//---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- set compare variation  
    array.Delta(0.001);  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

## Reserve

Allocates memory to increase the size of the array.

```
bool Reserve (  
    int size      // Number  
)
```

### Parameters

*size*

[in] The number of additional elements of the array.

### Return Value

true if successful, false - if there was an attempt to seek the amount is less than or equal to zero, or if the array did not increase.

### Note

To reduce memory fragmentation, increase the size of the array is made with a step previously given through the method of Step (int), or 16 (default).

### Example:

```
///--- example for CArrayFloat::Reserve(int)  
#include <Arrays\ArrayFloat.mqh>  
///---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- reserve memory  
    if(!array.Reserve(1024))  
    {  
        printf("Reserve error");  
        delete array;  
        return;  
    }  
    ///--- use array  
    ///--- . . .  
    ///--- delete array  
    delete array;  
}
```



## Resize

Sets a new (smaller) size of the array.

```
bool Resize(  
    int size      // Size  
)
```

### Parameters

*size*

[in] New size of the array.

### Return Value

true if successful, false - if there was an attempt to set the size of less than or equal to zero.

### Note

Changing the size of the array allows optimal use of memory. Superfluous elements on the right lost. To reduce fragmentation of memory, change the size of the array is made with a step previously given through the method of Step (int), or 16 (default).

### Example:

```
///--- example for CArrayFloat::Resize(int)  
#include <Arrays\ArrayFloat.mqh>  
///---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    ///--- resize array  
    if(!array.Resize(10))  
    {  
        printf("Resize error");  
        delete array;  
        return;  
    }  
    ///--- delete array  
    delete array;  
}
```

## Shutdown

Clears the array with a full memory release.

```
bool Shutdown()
```

### Return Value

true if successful, false - if an error occurred.

### Example:

```
//--- example for CArrayFloat::Shutdown()
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
{
    CArrayFloat *array=new CArrayFloat;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- add arrays elements
    //--- . . .
    //--- shutdown array
    if(!array.Shutdown())
    {
        printf("Shutdown error");
        delete array;
        return;
    }
    //--- delete array
    delete array;
}
```

## Add

Adds an element to the end of the array.

```
bool Add(  
    float element    // Element to add  
)
```

### Parameters

*element*

[in] Value of the element to add to the array.

### Return Value

true if successful, false - if you can not add an element.

### Example:

```
///--- example for CArrayFloat::Add(float)  
#include <Arrays\ArrayFloat.mqh>  
///---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    for(int i=0;i<100;i++)  
    {  
        if(!array.Add(i))  
        {  
            printf("Element addition error");  
            delete array;  
            return;  
        }  
    }  
    ///--- use array  
    ///--- . . .  
    ///--- delete array  
    delete array;  
}
```

## AddArray

Adds to the end of the array elements from another array.

```
bool AddArray(  
    const float& src[]      // Source array  
)
```

### Parameters

*src[]*

[in] Reference to an array of source elements to add.

### Return Value

true if successful, false - if you can not add items.

### Example:

```
///--- example for CArrayFloat::AddArray(const float &[])  
#include <Arrays\ArrayFloat.mqh>  
///---  
float src[];  
///---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add another array  
    if(!array.AddArray(src))  
    {  
        printf("Array addition error");  
        delete array;  
        return;  
    }  
    ///--- use array  
    ///--- . . .  
    ///--- delete array  
    delete array;  
}
```

## AddArray

Adds to the end of the array elements from another array.

```
bool AddArray(  
    const CArrayFloat* src      // Pointer to the source  
)
```

### Parameters

*src*

[in] Pointer to an instance of class CArrayFloat-source elements to add.

### Return Value

true if successful, false - if you can not add items.

### Example:

```
///--- example for CArrayFloat::AddArray(const CArrayFloat*)  
#include <Arrays\ArrayFloat.mqh>  
///---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- create source array  
    CArrayFloat *src=new CArrayFloat;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    ///--- add source arrays elements  
    ///--- . . .  
    ///--- add another array  
    if(!array.AddArray(src))  
    {  
        printf("Array addition error");  
        delete src;  
        delete array;  
        return;  
    }  
    ///--- delete source array  
    delete src;
```

```
//--- use array  
//--- . . .  
//--- delete array  
delete array;  
}
```

## Insert

Inserts an element in the array to the specified position.

```
bool Insert(  
    float element,    // Element to insert  
    int pos           // Position  
)
```

### Parameters

*element*

[in] Value of the element to be inserted into an array

*pos*

[in] Position in the array to insert

### Return Value

true if successful, false - if you can not insert the element.

### Example:

```
//--- example for CArrayFloat::Insert(float,int)  
#include <Arrays\ArrayFloat.mqh>  
//---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- insert elements  
    for(int i=0;i<100;i++)  
    {  
        if(!array.Insert(i,0))  
        {  
            printf("Insert error");  
            delete array;  
            return;  
        }  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

## InsertArray

Inserts an array of elements from another array with the specified position.

```
bool  InsertArray(  
    const float&  src[],      // Source array  
    int           pos         // Position  
)
```

### Parameters

*src[]*

[in] Reference to an array of source elements to insert

*pos*

[in] Position in the array to insert

### Return Value

true if successful, false - if you can not paste items.

### Example:

```
//--- example for CArrayFloat::InsertArray(const float &[],int)  
#include <Arrays\ArrayFloat.mqh>  
//---  
float src[];  
//---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- insert another array  
    if(!array.InsertArray(src,0))  
    {  
        printf("Array inserting error");  
        delete array;  
        return;  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```



## InsertArray

Inserts an array of elements from another array specified position.

```
bool InsertArray(  
    CArrayFloat* src,      // Pointer to the source  
    int pos               // Position  
)
```

### Parameters

*src*

[in] Pointer to an instance of class CArrayFloat-source elements to insert.

*pos*

[in] Position in the array to insert

### Return Value

true if successful, false - if you can not paste items.

### Example:

```
//--- example for CArrayFloat::InsertArray(const CArrayFloat*,int)  
#include <Arrays\ArrayFloat.mqh>  
//---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source array  
    CArrayFloat *src=new CArrayFloat;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    //--- add source arrays elements  
    //--- . . .  
    //--- insert another array  
    if(!array.InsertArray(src,0))  
    {  
        printf("Array inserting error");  
        delete src;  
        delete array;  
    }
```

```
        return;  
    }  
    //--- delete source array  
    delete src;  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

## AssignArray

Copies the array elements from another array.

```
bool AssignArray(  
    const float& src[]      // Source array  
)
```

### Parameters

*src[]*

[in] Reference to an array of source elements to copy.

### Return Value

true if successful, false - if you can not copy the items.

### Example:

```
///--- example for CArrayFloat::AssignArray(const float &[])  
#include <Arrays\ArrayFloat.mqh>  
///---  
float src[];  
///---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- assign another array  
    if(!array.AssignArray(src))  
    {  
        printf("Array assigned error");  
        delete array;  
        return;  
    }  
    ///--- use array  
    ///--- . . .  
    ///--- delete array  
    delete array;  
}
```

## AssignArray

Copies the array elements from another array.

```
bool AssignArray(  
    const CArrayFloat* src      // Pointer to the source  
)
```

### Parameters

*src*

[in] Pointer to an instance of class CArrayFloat-source element to copy.

### Return Value

true if successful, false - if you can not copy the items.

### Example:

```
///--- example for CArrayFloat::AssignArray(const CArrayFloat*)  
#include <Arrays\ArrayFloat.mqh>  
///---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- create source array  
    CArrayFloat *src =new CArrayFloat;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    ///--- add source arrays elements  
    ///--- . . .  
    ///--- assign another array  
    if(!array.AssignArray(src))  
    {  
        printf("Array assigned error");  
        delete src;  
        delete array;  
        return;  
    }  
    ///--- arrays is identical  
    ///--- delete source array
```

```
delete src;  
//--- use array  
//--- . . .  
//--- delete array  
delete array;  
}
```

## Update

Changes the element at the specified position array.

```
bool Update(  
    int    pos,          // Position  
    float  element       // Value  
)
```

### Parameters

*pos*

[in] Position of the element in the array to change

*element*

[in] New value element

### Return Value

true if successful, false - if you can not change the element.

### Example:

```
//--- example for CArrayFloat::Update(int,float)  
#include <Arrays\ArrayFloat.mqh>  
//---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- update element  
    if(!array.Update(0,100.0))  
    {  
        printf("Update error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

## Shift

Moves an item from a given position in the array to the specified offset.

```
bool Shift(  
    int pos,          // Position  
    int shift         // Shift  
)
```

### Parameters

*pos*

[in] Position of the moved element array

*shift*

[in] The value of displacement (both positive and negative).

### Return Value

true if successful, false - if you can not move the item.

### Example:

```
//--- example for CArrayFloat::Shift(int,int)  
#include <Arrays\ArrayFloat.mqh>  
//---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- shift element  
    if(!array.Shift(10,-5))  
    {  
        printf("Shift error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

## Delete

Removes the element from the given position in the array.

```
bool Delete(  
    int pos    // Position  
)
```

### Parameters

*pos*

[in] Position removes the element in the array.

### Return Value

true if successful, false - if you can not remove the element.

### Example:

```
///--- example for CArrayFloat::Delete(int)  
#include <Arrays\ArrayFloat.mqh>  
///---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    ///--- delete element  
    if(!array.Delete(0))  
    {  
        printf("Delete error");  
        delete array;  
        return;  
    }  
    ///--- delete array  
    delete array;  
}
```



## DeleteRange

Deletes a group of elements from a given position in the array.

```
bool DeleteRange(  
    int from,      // Position of the first element  
    int to         // Position of last element  
)
```

### Parameters

*from*

[in] Position of the first removes the element in the array.

*to*

[in] Position of the last deleted element in the array.

### Return Value

true if successful, false - if you can not remove elements.

### Example:

```
//--- example for CArrayFloat::DeleteRange(int,int)  
#include <Arrays\ArrayFloat.mqh>  
//---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- delete elements  
    if(!array.DeleteRange(0,10))  
    {  
        printf("Delete error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

## At

Gets the element from the given position in the array.

```
float At(  
    int pos    // Position  
) const
```

### Parameters

*pos*

[in] Position of the desired element in the array.

### Return Value

The value of the element in case of success, FLT\_MAX if there was an attempt to get an element of not existing positions (the last error ERR\_OUT\_OF\_RANGE).

### Note

Of course, FLT\_MAX may be a valid value of an array element, so having a value, always check the last error code.

### Example:

```
//--- example for CArrayFloat::At(int)  
#include <Arrays\ArrayFloat.mqh>  
//---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    for(int i=0;i<array.Total();i++)  
    {  
        float result=array.At(i);  
        if(result==FLT_MAX && GetLastError()==ERR_OUT_OF_RANGE)  
        {  
            //--- error reading from array  
            printf("Get element error");  
            delete array;  
            return;  
        }  
        //--- use element  
        //--- . . .  
    }  
}
```

```
//--- delete array  
delete array;  
}
```

## CompareArray

Compares array with another array.

```
bool CompareArray(  
    const float& src[]      // Source array  
    ) const
```

### Parameters

*src[]*

[in] Reference to an array of source elements for comparison.

### Return Value

true if arrays are equal, false - if not.

### Example:

```
//--- example for CArrayFloat::CompareArray(const float &[])  
#include <Arrays\ArrayFloat.mqh>  
//---  
float src[];  
//---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- compare with another array  
    int result=array.CompareArray(src);  
    //--- delete array  
    delete array;  
}
```

## AssignArrayconst

Copies the array elements from another array.

```
bool AssignArrayconst(  
    const CArrayFloat* src      // Pointer to the source  
    ) const
```

### Parameters

*src*

[in] Pointer to an instance of class CArrayFloat-source element to copy.

### Return Value

true if successful, false - if you can not copy the items.

### Example:

```
///--- example for CArrayFloat::CompareArray(const CArrayFloat*)  
#include <Arrays\ArrayFloat.mqh>  
///---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- create source array  
    CArrayFloat *src=new CArrayFloat;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    ///--- add source arrays elements  
    ///--- . . .  
    ///--- compare with another array  
    int result=array.CompareArray(src);  
    ///--- delete arrays  
    delete src;  
    delete array;  
}
```

## InsertSort

Inserts element in a sorted array.

```
bool InsertSort(  
    float element    // Element to insert  
)
```

### Parameters

*element*

[in] Value of the element to be inserted into a sorted array

### Return Value

true if successful, false - if you can not insert the element.

### Example:

```
///--- example for CArrayFloat::InsertSort(float)  
#include <Arrays\ArrayFloat.mqh>  
///---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    ///--- sort array  
    array.Sort();  
    ///--- insert element  
    if(!array.InsertSort(100.0))  
    {  
        printf("Insert error");  
        delete array;  
        return;  
    }  
    ///--- delete array  
    delete array;  
}
```

## Search

Searches for an element equal to the sample in the sorted array.

```
int Search(  
    float element    // Sample  
    ) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayFloat::Search(float)  
#include <Arrays\ArrayFloat.mqh>  
//---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.Search(100.0)!=-1) printf("Element found");  
    else                        printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchGreat

Searches for an element of more samples in sorted array.

```
int SearchGreat(  
    float element    // Sample  
    ) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayFloat::SearchGreat(float)  
#include <Arrays\ArrayFloat.mqh>  
//---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchGreat(100.0)!=-1) printf("Element found");  
    else                             printf("Element not found");  
    //--- delete array  
    delete array;  
}
```



## SearchLess

Searches for an element less than the sample in the sorted array.

```
int SearchLess(  
    float element    // Sample  
    ) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayFloat:: SearchLess(float)  
#include <Arrays\ArrayFloat.mqh>  
//---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchLess(100.0)!=-1) printf("Element found");  
    else                          printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchGreatOrEqual

Searches for an element greater than or equal to the sample in the sorted array.

```
int SearchGreatOrEqual(  
    float element    // Sample  
) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayFloat::SearchGreatOrEqual(float)  
#include <Arrays\ArrayFloat.mqh>  
//---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchGreatOrEqual(100.0)!=-1) printf("Element found");  
    else                                  printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchLessOrEqual

Searches for an element less than or equal to the sample in the sorted array.

```
int SearchLessOrEqual(  
    float element    // Sample  
) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayFloat::SearchLessOrEqual(float)  
#include <Arrays\ArrayFloat.mqh>  
//---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchLessOrEqual(100.0)!=-1) printf("Element found");  
    else                                printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchFirst

Finds the first element equal to the sample in the sorted array.

```
int SearchFirst(  
    float element    // Sample  
    ) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayFloat::SearchFirst(float)  
#include <Arrays\ArrayFloat.mqh>  
//---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchFirst(100.0)!=-1) printf("Element found");  
    else                             printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchLast

Finds the last element equal to the model in sorted array.

```
int SearchLast(  
    float element    // Sample  
    ) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayFloat::SearchLast(float)  
#include <Arrays\ArrayFloat.mqh>  
//---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchLast(100.0)!=-1) printf("Element found");  
    else                             printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchLinear

Searches for the element equal to the sample in the array.

```
int SearchLinear(  
    float element    // Sample  
) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the element was not found.

### Note

The method uses the linear search (or sequential search) algorithm for unsorted arrays.

### Example:

```
///--- example for CArrayFloat::SearchLinear(float)  
#include <Arrays\ArrayFloat.mqh>  
///---  
void OnStart()  
{  
    CArrayFloat *array=new CArrayFloat;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    ///--- search element  
    if(array.SearchLinear(100.0) != -1) printf("Element found");  
    else                               printf("Element not found");  
    ///--- delete array  
    delete array;  
}
```

## Save

Saves data array in the file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] Handle to open earlier, with the function FileOpen (...), binary file.

### Return Value

true - if successfully completed, false - if an error.

### Example:

```
//--- example for CArrayFloat::Save(int)  
#include <Arrays\ArrayFloat.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array!=NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add 100 arrays elements  
    for(int i=0;i<100;i++)  
    {  
        array.Add(i);  
    }  
    //--- open file  
    file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!array.Save(file_handle))  
        {  
            //--- file save error  
            printf("File save: Error %d!",GetLastError());  
            delete array;  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }
```

```
    }  
    delete array;  
}
```



## Load

Loads data array from a file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] Handle to open earlier, with the function FileOpen (...), binary file.

### Return Value

true - if successfully completed, false - if an error.

### Example:

```
//--- example for CArrayFloat::Load(int)  
#include <Arrays\ArrayFloat.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CArrayFloat *array=new CArrayFloat;  
    //---  
    if(array!=NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- open file  
    file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!array.Load(file_handle))  
        {  
            //--- file load error  
            printf("File load: Error %d!",GetLastError());  
            delete array;  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }  
    //--- use arrays elements  
    for(int i=0;i<array.Total();i++)  
    {  
        printf("Element[%d] = %f",i,array.At(i));  
    }  
}
```

```
    }  
    delete array;  
}
```

## Type

Gets the type identifier of the array.

```
virtual int Type() const
```

### Return Value

ID type of the array (for CArrayFloat - 87).

### Example:

```
//--- example for CArrayFloat::Type()
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
{
    CArrayFloat *array=new CArrayFloat;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- get array type
    int type=array.Type();
    //--- delete array
    delete array;
}
```

## CArrayDouble

CArrayDouble class is a class of dynamic array of variables of type double.

### Description

Class CArrayDouble provides the ability to work with a dynamic array of variables of type double. In the class implemented the ability to add / insert / delete elements in an array, sort array, searching in sorted array. In addition, the implemented methods of work with the file.

### Declaration

```
class CArrayDouble : public CArray
```

### Title

```
#include <Arrays\ArrayDouble.mqh>
```

### Class Methods

| Attributes                  |                                                                             |
|-----------------------------|-----------------------------------------------------------------------------|
| <a href="#">Delta</a>       | Set the comparison tolerance                                                |
| Memory control              |                                                                             |
| <a href="#">Reserve</a>     | Allocates memory to increase the size of the array                          |
| <a href="#">Resize</a>      | Sets a new (smaller) size of the array                                      |
| <a href="#">Shutdown</a>    | Clears the array with a full memory release                                 |
| Add methods                 |                                                                             |
| <a href="#">Add</a>         | Adds an element to the end of the array                                     |
| <a href="#">AddArray</a>    | Adds to the end of the array elements from another array                    |
| <a href="#">AddArray</a>    | Adds to the end of the array elements from another array                    |
| <a href="#">Insert</a>      | Inserts an element in the array to the specified position                   |
| <a href="#">InsertArray</a> | Inserts an array of elements from another array with the specified position |
| <a href="#">InsertArray</a> | Inserts an array of elements from another array with the specified position |
| <a href="#">AssignArray</a> | Copies the array elements from another array                                |
| <a href="#">AssignArray</a> | Copies the array elements from another array                                |
| Update methods              |                                                                             |

|                                    |                                                                            |
|------------------------------------|----------------------------------------------------------------------------|
| <a href="#">Update</a>             | Changes the element at the specified position array                        |
| <a href="#">Shift</a>              | Moves an item from a given position in the array to the specified offset   |
| <b>Delete methods</b>              |                                                                            |
| <a href="#">Delete</a>             | Removes the element from the specified position array                      |
| <a href="#">DeleteRange</a>        | Deletes a group of elements from the specified position array              |
| <b>Access methods</b>              |                                                                            |
| <a href="#">At</a>                 | Gets the element from the specified position array                         |
| <b>Compare methods</b>             |                                                                            |
| <a href="#">CompareArray</a>       | Compares array with another array                                          |
| <a href="#">CompareArray</a>       | Compares array with another array                                          |
| <b>Search for min/max</b>          |                                                                            |
| <a href="#">Minimum</a>            | Gets index of the lowest element of the array in the specified range       |
| <a href="#">Maximum</a>            | Gets index of the highest element of the array in the specified range      |
| <b>Sorted array operations</b>     |                                                                            |
| <a href="#">InsertSort</a>         | Inserts element in a sorted array                                          |
| <a href="#">Search</a>             | Searches for an element equal to the model in sorted array                 |
| <a href="#">SearchGreat</a>        | Searches for an element of more samples in sorted array                    |
| <a href="#">SearchLess</a>         | Searches for an element less than the sample in the sorted array           |
| <a href="#">SearchGreatOrEqual</a> | Searches for an element greater than or equal to the model in sorted array |
| <a href="#">SearchLessOrEqual</a>  | Searches for an element less than or equal to the model in sorted array    |
| <a href="#">SearchFirst</a>        | Finds the first element equal to the model in sorted array                 |
| <a href="#">SearchLast</a>         | Finds the last element equal to the model in sorted array                  |
| <a href="#">SearchLinear</a>       | Searches for the element equal to the sample in the array                  |

| Input/output                 |                                       |
|------------------------------|---------------------------------------|
| virtual <a href="#">Save</a> | Saves data array in the file          |
| virtual <a href="#">Load</a> | Loads data array from a file          |
| virtual <a href="#">Type</a> | Gets the type identifier of the array |

**Derived classes:**

- [CIndicatorBuffer](#)

## Delta

Set tolerance comparison.

```
void Delta(  
    double delta    // Tolerance  
)
```

### Parameters

*delta*

[in] The new value of the admission of comparison.

### Return Value

No

### Note

Admission of comparison used in the search. Values are considered equal if their difference is less than or equal to tolerance. The default tolerance is 0.0.

### Example:

```
///--- example for CArrayDouble::Delta(double)  
#include <Arrays\ArrayDouble.mqh>  
///---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- set compare variation  
    array.Delta(0.001);  
    ///--- use array  
    ///--- . . .  
    ///--- delete array  
    delete array;  
}
```

## Reserve

Allocates memory to increase the size of the array.

```
bool Reserve (  
    int size      // Number  
)
```

### Parameters

*size*

[in] The number of additional elements of the array.

### Return Value

true if successful, false - if there was an attempt to seek the amount is less than or equal to zero, or if the array did not increase.

### Note

To reduce memory fragmentation, increase the size of the array is made with a step previously given through the method of Step (int), or 16 (default).

### Example:

```
//--- example for CArrayDouble::Reserve(int)  
#include <Arrays\ArrayDouble.mqh>  
//---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- reserve memory  
    if(!array.Reserve(1024))  
    {  
        printf("Reserve error");  
        delete array;  
        return;  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```



## Resize

Sets a new (smaller) size of the array.

```
bool  Resize(  
    int  size      // Size  
)
```

### Parameters

*size*

[in] New size of the array.

### Return Value

true if successful, false - if there was an attempt to set the size of less than or equal to zero.

### Note

Changing the size of the array allows optimal use of memory. Superfluous elements on the right lost. To reduce fragmentation of memory, change the size of the array is made with a step previously given through the method of Step (int), or 16 (default).

### Example:

```
///--- example for CArrayDouble::Resize(int)  
#include <Arrays\ArrayDouble.mqh>  
///---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    ///--- resize array  
    if(!array.Resize(10))  
    {  
        printf("Resize error");  
        delete array;  
        return;  
    }  
    ///--- delete array  
    delete array;  
}
```

## Shutdown

Clears the array with a full memory release.

```
bool Shutdown()
```

### Return Value

true if successful, false - if an error occurred.

### Example:

```
//--- example for CArrayDouble::Shutdown()
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
{
    CArrayDouble *array=new CArrayDouble;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- add arrays elements
    //--- . . .
    //--- shutdown array
    if(!array.Shutdown())
    {
        printf("Shutdown error");
        delete array;
        return;
    }
    //--- delete array
    delete array;
}
```

## Add

Adds an element to the end of the array.

```
bool Add(  
    double element    // Element to add  
)
```

### Parameters

*element*

[in] value of the element to add to the array.

### Return Value

true if successful, false - if you can not add an element.

### Example:

```
///--- example for CArrayDouble::Add(double)  
#include <Arrays\ArrayDouble.mqh>  
///---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    for(int i=0;i<100;i++)  
    {  
        if(!array.Add(i))  
        {  
            printf("Element addition error");  
            delete array;  
            return;  
        }  
    }  
    ///--- use array  
    ///--- . . .  
    ///--- delete array  
    delete array;  
}
```

## AddArray

Adds to the end of the array elements from another array.

```
bool AddArray(  
    const double& src[]      // Source array  
)
```

### Parameters

*src[]*

[in] Reference to an array of source elements to add.

### Return Value

true if successful, false - if you can not add items.

### Example:

```
///--- example for CArrayDouble::AddArray(const double &[])  
#include <Arrays\ArrayDouble.mqh>  
///---  
double src[];  
///---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add another array  
    if(!array.AddArray(src))  
    {  
        printf("Array addition error");  
        delete array;  
        return;  
    }  
    ///--- use array  
    ///--- . . .  
    ///--- delete array  
    delete array;  
}
```

## AddArray

Adds to the end of the array elements from another array.

```
bool AddArray(  
    const CArrayDouble* src      // Pointer to the source  
)
```

### Parameters

*src*

[in] Pointer to an instance of class CArrayDouble-source elements to add.

### Return Value

true if successful, false - if you can not add items.

### Example:

```
///--- example for CArrayDouble::AddArray(const CArrayDouble*)  
#include <Arrays\ArrayDouble.mqh>  
///---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- create source array  
    CArrayDouble *src=new CArrayDouble;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    ///--- add source arrays elements  
    ///--- . . .  
    ///--- add another array  
    if(!array.AddArray(src))  
    {  
        printf("Array addition error");  
        delete src;  
        delete array;  
        return;  
    }  
    ///--- delete source array  
    delete src;
```

```
//--- use array  
//--- . . .  
//--- delete array  
delete array;  
}
```

## Insert

Inserts an element in the array to the specified position.

```
bool Insert(  
    double element,    // Element to insert  
    int pos            // Position  
)
```

### Parameters

*element*

[in] Value of the element to be inserted into an array

*pos*

[in] Position in the array to insert

### Return Value

true if successful, false - if you can not insert the element.

### Example:

```
//--- example for CArrayDouble::Insert(double,int)  
#include <Arrays\ArrayDouble.mqh>  
//---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- insert elements  
    for(int i=0;i<100;i++)  
    {  
        if(!array.Insert(i,0))  
        {  
            printf("Insert error");  
            delete array;  
            return;  
        }  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

## InsertArray

Inserts an array of elements from another array specified position.

```
bool InsertArray(  
    const double& src[],      // Source array  
    int pos                  // Position  
)
```

### Parameters

*src[]*

[in] Reference to an array of source elements to insert

*pos*

[in] Position in the array to insert

### Return Value

true if successful, false - if you can not paste items.

### Example:

```
//--- example for CArrayDouble::InsertArray(const double &[],int)  
#include <Arrays\ArrayDouble.mqh>  
//---  
double src[];  
//---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- insert another array  
    if(!array.InsertArray(src,0))  
    {  
        printf("Array inserting error");  
        delete array;  
        return;  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```



## InsertArray

Inserts an array of elements from another array specified position.

```
bool  InsertArray(  
    CArrayDouble*  src,      // Pointer to the source  
    int            pos       // Position  
)
```

### Parameters

*src*

[in] Pointer to an instance of class CArrayDouble-source elements to insert.

*pos*

[in] Position in the array to insert

### Return Value

true if successful, false - if you can not paste items.

### Example:

```
//--- example for CArrayDouble::InsertArray(const CArrayDouble*,int)  
#include <Arrays\ArrayDouble.mqh>  
//---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source array  
    CArrayDouble *src=new CArrayDouble;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    //--- add source arrays elements  
    //--- . . .  
    //--- insert another array  
    if(!array.InsertArray(src,0))  
    {  
        printf("Array inserting error");  
        delete src;  
        delete array;  
    }  
}
```

```
        return;  
    }  
    //--- delete source array  
    delete src;  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

## AssignArray

Copies the array elements from another array.

```
bool AssignArray(  
    const double& src[]      // Source array  
)
```

### Parameters

*src[]*

[in] Reference to an array of source elements to copy.

### Return Value

true if successful, false - if you can not copy the items.

### Example:

```
///--- example for CArrayDouble::AssignArray(const double &[])  
#include <Arrays\ArrayDouble.mqh>  
///---  
double src[];  
///---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- assign another array  
    if(!array.AssignArray(src))  
    {  
        printf("Array assigned error");  
        delete array;  
        return;  
    }  
    ///--- use array  
    ///--- . . .  
    ///--- delete array  
    delete array;  
}
```

## AssignArray

Copies the array elements from another array.

```
bool AssignArray(  
    const CArrayDouble* src      // Pointer to the source  
)
```

### Parameters

*src*

[in] Pointer to an instance of class CArrayDouble-source element to copy.

### Return Value

true if successful, false - if you can not copy the items.

### Example:

```
///--- example for CArrayDouble::AssignArray(const CArrayDouble*)  
#include <Arrays\ArrayDouble.mqh>  
///---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- create source array  
    CArrayDouble *src =new CArrayDouble;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    ///--- add source arrays elements  
    ///--- . . .  
    ///--- assign another array  
    if(!array.AssignArray(src))  
    {  
        printf("Array assigned error");  
        delete src;  
        delete array;  
        return;  
    }  
    ///--- arrays is identical  
    ///--- delete source array
```

```
delete src;  
//--- use array  
//--- . . .  
//--- delete array  
delete array;  
}
```

## Update

Changes the element at the specified position array.

```
bool Update(  
    int    pos,           // Position  
    double element       // Value  
)
```

### Parameters

*pos*

[in] Position of the element in the array to change

*element*

[in] New value.

### Return Value

true if successful, false - if you can not change the element.

### Example:

```
//--- example for CArrayDouble::Update(int,double)  
#include <Arrays\ArrayDouble.mqh>  
//---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- update element  
    if(!array.Update(0,100.0))  
    {  
        printf("Update error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

## Shift

Moves an item from a given position in the array to the specified offset.

```
bool Shift(  
    int pos,          // Position  
    int shift         // Shift  
)
```

### Parameters

*pos*

[in] Position of the moved element array

*shift*

[in] The value of displacement (both positive and negative).

### Return Value

true if successful, false - if you can not move the item.

### Example:

```
//--- example for CArrayDouble::Shift(int,int)  
#include <Arrays\ArrayDouble.mqh>  
//---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- shift element  
    if(!array.Shift(10,-5))  
    {  
        printf("Shift error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

## Delete

Removes the element from the given position in the array.

```
bool Delete(  
    int pos    // Position  
)
```

### Parameters

*pos*

[in] position removes the element in the array.

### Return Value

true if successful, false - if you can not remove the element.

### Example:

```
///--- example for CArrayDouble::Delete(int)  
#include <Arrays\ArrayDouble.mqh>  
///---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    ///--- delete element  
    if(!array.Delete(0))  
    {  
        printf("Delete error");  
        delete array;  
        return;  
    }  
    ///--- delete array  
    delete array;  
}
```



## DeleteRange

Deletes a group of elements from a given position in the array.

```
bool DeleteRange(  
    int from,      // Position of the first element  
    int to         // Position of last element  
)
```

### Parameters

*from*

[in] Position of the first removes the element in the array.

*to*

[in] Position of the last deleted element in the array.

### Return Value

true if successful, false - if you can not remove elements.

### Example:

```
//--- example for CArrayDouble::DeleteRange(int,int)  
#include <Arrays\ArrayDouble.mqh>  
//---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- delete elements  
    if(!array.DeleteRange(0,10))  
    {  
        printf("Delete error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

## At

Gets the element from the given position in the array.

```
double At(  
    int pos      // Position  
) const
```

### Parameters

*pos*

[in] Position of the desired element in the array.

### Return Value

The value of the element in case of success, DBL\_MAX if there was an attempt to get an element of not existing positions (the last error ERR\_OUT\_OF\_RANGE).

### Note

Of course, DBL\_MAX may be a valid value of an array element, so having a value, always check the last error code.

### Example:

```
//--- example for CArrayDouble::At(int)  
#include <Arrays\ArrayDouble.mqh>  
//---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    for(int i=0;i<array.Total();i++)  
    {  
        double result=array.At(i);  
        if(result==DBL_MAX && GetLastError()==ERR_OUT_OF_RANGE)  
        {  
            //--- Error reading from the array  
            printf("Get element error");  
            delete array;  
            return;  
        }  
        //--- use element  
        //--- . . .  
    }  
}
```

```
//--- delete array  
delete array;  
}
```

## CompareArray

Compares array with another array.

```
bool CompareArray(  
    const double& src[]      // Source array  
    ) const
```

### Parameters

*src[]*

[in] Reference to an array of source elements for comparison.

### Return Value

true if arrays are equal, false - if not.

### Example:

```
///--- example for CArrayDouble::CompareArray(const double &[])  
#include <Arrays\ArrayDouble.mqh>  
///---  
double src[];  
///---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- compare with another array  
    int result=array.CompareArray(src);  
    ///--- delete array  
    delete array;  
}
```

## CompareArray

Compares array with another array.

```
bool CompareArray(  
    const CArrayDouble* src      // Pointer to the source  
    ) const
```

### Parameters

*src*

[in] Pointer to an instance of class CArrayDouble-source elements for comparison.

### Return Value

true if arrays are equal, false - if not.

### Example:

```
//--- example for CArrayDouble::CompareArray(const CArrayDouble*)  
#include <Arrays\ArrayDouble.mqh>  
//---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source array  
    CArrayDouble *src=new CArrayDouble;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    //--- add source arrays elements  
    //--- . . .  
    //--- compare with another array  
    int result=array.CompareArray(src);  
    //--- delete arrays  
    delete src;  
    delete array;  
}
```

## Minimum

Gets index of the lowest element of the array in the specified range.

```
int Minimum(  
    int start,      // starting index  
    int count       // number of elements to proceed  
) const
```

### Parameters

*start*

[in] Starting index of the array.

*count*

[in] Number of elements to proceed.

### Returned value

Index of the lowest element of the array in the specified range.

## Maximum

Gets index of the highest element of the array in the specified range.

```
int Maximum(  
    int start,      // starting index  
    int count       // number of elements to proceed  
) const
```

### Parameters

*start*

[in] Starting index of the array.

*count*

[in] Number of elements to proceed.

### Returned value

Index of the highest element of the array in the specified range.

## InsertSort

Inserts element in a sorted array.

```
bool InsertSort(  
    double element    // Element to insert  
)
```

### Parameters

*element*

[in] value of the element to be inserted into a sorted array

### Return Value

true if successful, false - if you can not insert the element.

### Example:

```
///--- example for CArrayDouble::InsertSort(double)  
#include <Arrays\ArrayDouble.mqh>  
///---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    ///--- sort array  
    array.Sort();  
    ///--- insert element  
    if(!array.InsertSort(100.0))  
    {  
        printf("Insert error");  
        delete array;  
        return;  
    }  
    ///--- delete array  
    delete array;  
}
```



## Search

Searches for an element equal to the sample in the sorted array.

```
int Search(  
    double element    // Sample  
    ) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayDouble::Search(double)  
#include <Arrays\ArrayDouble.mqh>  
//---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.Search(100.0)!=-1) printf("Element found");  
    else                        printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchGreat

Searches for an element of more samples in sorted array.

```
int SearchGreat(  
    double element    // Sample  
    ) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayDouble::SearchGreat(double)  
#include <Arrays\ArrayDouble.mqh>  
//---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchGreat(100.0)!=-1) printf("Element found");  
    else                             printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchLess

Searches for an element less than the sample in the sorted array.

```
int SearchLess(  
    double element    // Sample  
    ) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayDouble:: SearchLess(double)  
#include <Arrays\ArrayDouble.mqh>  
//---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchLess(100.0)!=-1) printf("Element found");  
    else                          printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchGreatOrEqual

Searches for an element greater than or equal to the sample in the sorted array.

```
int SearchGreatOrEqual(  
    double element    // Sample  
) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
///--- example for CArrayDouble::SearchGreatOrEqual(double)  
#include <Arrays\ArrayDouble.mqh>  
///---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    ///--- sort array  
    array.Sort();  
    ///--- search element  
    if(array.SearchGreatOrEqual(100.0)!=-1) printf("Element found");  
    else                                  printf("Element not found");  
    ///--- delete array  
    delete array;  
}
```

## SearchLessOrEqual

Searches for an element less than or equal to the sample in the sorted array.

```
int SearchLessOrEqual(  
    double element    // Sample  
) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
///--- example for CArrayDouble::SearchLessOrEqual(double)  
#include <Arrays\ArrayDouble.mqh>  
///---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    ///--- sort array  
    array.Sort();  
    ///--- search element  
    if(array.SearchLessOrEqual(100.0)!=-1) printf("Element found");  
    else                                printf("Element not found");  
    ///--- delete array  
    delete array;  
}
```

## SearchFirst

Finds the first element equal to the sample in the sorted array.

```
int SearchFirst(  
    double element    // Sample  
    ) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayDouble::SearchFirst(double)  
#include <Arrays\ArrayDouble.mqh>  
//---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchFirst(100.0)!=-1) printf("Element found");  
    else                             printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchLast

Finds the last element equal to the model in sorted array.

```
int SearchLast(  
    double element    // Sample  
    ) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayDouble::SearchLast(double)  
#include <Arrays\ArrayDouble.mqh>  
//---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchLast(100.0)!=-1) printf("Element found");  
    else                           printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchLinear

Searches for the element equal to the sample in the sorted array.

```
int SearchLinear(  
    double element    // Sample  
) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the element was not found.

### Note

The method uses the linear search (or sequential search) algorithm for unsorted arrays.

### Example:

```
///--- example for CArrayDouble::SearchLinear(double)  
#include <Arrays\ArrayDouble.mqh>  
///---  
void OnStart()  
{  
    CArrayDouble *array=new CArrayDouble;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    ///--- search element  
    if(array.SearchLinear(100.0) != -1) printf("Element found");  
    else                               printf("Element not found");  
    ///--- delete array  
    delete array;  
}
```



## Save

Saves data array in the file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] Handle to open earlier, with the function FileOpen (...), binary file.

### Return Value

true - if successfully completed, false - if an error.

### Example:

```
//--- example for CArrayDouble::Save(int)  
#include <Arrays\ArrayDouble.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CArrayDouble *array=new CArrayDouble;  
    //---  
    if(array!=NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add 100 arrays elements  
    for(int i=0;i<100;i++)  
    {  
        array.Add(i);  
    }  
    //--- open file  
    file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!array.Save(file_handle))  
        {  
            //--- file save error  
            printf("File save: Error %d!",GetLastError());  
            delete array;  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }
```

```
    }  
    ///--- delete array  
    delete array;  
}
```

## Load

Loads data array from a file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] Handle to open earlier, with the function FileOpen (...), binary file.

### Return Value

true - if successfully completed, false - if an error.

### Example:

```
//--- example for CArrayDouble::Load(int)  
#include <Arrays\ArrayDouble.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CArrayDouble *array=new CArrayDouble;  
    //---  
    if(array!=NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- open file  
    file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!array.Load(file_handle))  
        {  
            //--- file load error  
            printf("File load: Error %d!",GetLastError());  
            delete array;  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }  
    //--- use arrays elements  
    for(int i=0;i<array.Total();i++)  
    {  
        printf("Element[%d] = %f",i,array.At(i));  
    }  
}
```

```
    }  
    ///--- delete array  
    delete array;  
}
```

## Type

Gets the type identifier of the array.

```
virtual int Type() const
```

### Return Value

ID type of the array (for CArrayDouble - 87).

### Example:

```
//--- example for CArrayDouble::Type()
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
{
    CArrayDouble *array=new CArrayDouble;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- get array type
    int type=array.Type();
    //--- delete array
    delete array;
}
```

## CArrayString

CArrayString class is a class of dynamic array of variables of type string.

### Description

Class CArrayString provides the ability to work with a dynamic array of variables of type string. In the class implemented the ability to add / insert / delete elements in an array, sort array, searching in sorted array. In addition, the implemented methods of work with the file.

### Declaration

```
class CArrayString : public CArray
```

### Title

```
#include <Arrays\ArrayString.mqh>
```

### Class Methods

| Memory control              |                                                                             |
|-----------------------------|-----------------------------------------------------------------------------|
| <a href="#">Reserve</a>     | Allocates memory to increase the size of the array                          |
| <a href="#">Resize</a>      | Sets a new (smaller) size of the array                                      |
| <a href="#">Shutdown</a>    | Sets a new (smaller) size of the array                                      |
| Add methods                 |                                                                             |
| <a href="#">Add</a>         | Adds an element to the end of the array                                     |
| <a href="#">AddArray</a>    | Adds to the end of the array elements from another array                    |
| <a href="#">AddArray</a>    | Adds to the end of the array elements from another array                    |
| <a href="#">Insert</a>      | Inserts an element in the array to the specified position                   |
| <a href="#">InsertArray</a> | Inserts an array of elements from another array with the specified position |
| <a href="#">InsertArray</a> | Inserts an array of elements from another array with the specified position |
| <a href="#">AssignArray</a> | Copies the array elements from another array                                |
| <a href="#">AssignArray</a> | Copies the array elements from another array                                |
| Update methods              |                                                                             |
| <a href="#">Update</a>      | Changes the element at the specified position array                         |

|                                    |                                                                             |
|------------------------------------|-----------------------------------------------------------------------------|
| <a href="#">Shift</a>              | Moves an item from a given position in the array to the specified offset    |
| <b>Delete methods</b>              |                                                                             |
| <a href="#">Delete</a>             | Removes the element from the specified position array                       |
| <a href="#">DeleteRange</a>        | Deletes a group of elements from the specified position array               |
| <b>Access methods</b>              |                                                                             |
| <a href="#">At</a>                 | Gets the element from the specified position array                          |
| <b>Compare methods</b>             |                                                                             |
| <a href="#">CompareArray</a>       | Compares array with another array                                           |
| <a href="#">CompareArray</a>       | Compares array with another array                                           |
| <b>Sorted array opetations</b>     |                                                                             |
| <a href="#">InsertSort</a>         | Inserts element in a sorted array                                           |
| <a href="#">Search</a>             | Searches for an element equal to the sample in sorted array                 |
| <a href="#">SearchGreat</a>        | Searches for an element of more samples in sorted array                     |
| <a href="#">SearchLess</a>         | Searches for an element less than the sample in the sorted array            |
| <a href="#">SearchGreatOrEqual</a> | Searches for an element greater than or equal to the sample in sorted array |
| <a href="#">SearchLessOrEqual</a>  | Searches for an element less than or equal to the sample in sorted array    |
| <a href="#">SearchFirst</a>        | Finds the first element equal to the sample in sorted array                 |
| <a href="#">SearchLast</a>         | Finds the last element equal to the sample in sorted array                  |
| <a href="#">SearchLinear</a>       | Searches for the element equal to the sample in the array                   |
| <b>Input/output</b>                |                                                                             |
| virtual <a href="#">Save</a>       | Saves data array in the file                                                |
| virtual <a href="#">Load</a>       | Loads data array from a file                                                |
| virtual <a href="#">Type</a>       | Gets the type identifier of the array                                       |

## Reserve

Allocates memory to increase the size of the array.

```
bool Reserve (  
    int size      // Number  
)
```

### Parameters

*size*

[in] The number of additional elements of the array.

### Return Value

true if successful, false - if there was an attempt to seek the amount is less than or equal to zero, or if the array did not increase.

### Note

To reduce memory fragmentation, increase the size of the array is made with a step previously given through the method of Step (int), or 16 (default).

### Example:

```
//--- example for CArrayString::Reserve(int)  
#include <Arrays\ArrayString.mqh>  
//---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- reserve memory  
    if(!array.Reserve(1024))  
    {  
        printf("Reserve error");  
        delete array;  
        return;  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```



## Resize

Sets a new (smaller) size of the array.

```
bool  Resize(  
    int  size      // Size  
)
```

### Parameters

*size*

[in] New size of the array.

### Return Value

true if successful, false - if there was an attempt to set the size of less than or equal to zero.

### Note

Changing the size of the array allows optimal use of memory. Superfluous elements on the right lost. To reduce fragmentation of memory, change the size of the array is made with a step previously given through the method of Step (int), or 16 (default).

### Example:

```
/*--- example for CArrayString::Resize(int)  
#include <Arrays\ArrayString.mqh>  
/*---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    /*---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    /*--- add arrays elements  
    /*--- . . .  
    /*--- resize array  
    if(!array.Resize(10))  
    {  
        printf("Resize error");  
        delete array;  
        return;  
    }  
    /*--- delete array  
    delete array;  
}
```

## Shutdown

Clears the array with a full memory release.

```
bool Shutdown()
```

### Return Value

true if successful, false - if an error occurred.

### Example:

```
//--- example for CArrayString::Shutdown()
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
{
    CArrayString *array=new CArrayString;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- add arrays elements
    //--- . . .
    //--- shutdown array
    if(!array.Shutdown())
    {
        printf("Shutdown error");
        delete array;
        return;
    }
    //--- delete array
    delete array;
}
```

## Add

Adds an element to the end of the array.

```
bool Add(  
    string element    // Element to add  
)
```

### Parameters

*element*

[in] value of the element to add to the array.

### Return Value

true if successful, false - if you can not add an element.

### Example:

```
///--- example for CArrayString::Add(string)  
#include <Arrays\ArrayString.mqh>  
///---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    for(int i=0;i<100;i++)  
    {  
        if(!array.Add(IntegerToString(i)))  
        {  
            printf("Element addition error");  
            delete array;  
            return;  
        }  
    }  
    ///--- use array  
    ///--- . . .  
    ///--- delete array  
    delete array;  
}
```

## AddArray

Adds to the end of the array elements from another array.

```
bool AddArray(  
    const string& src[]      // Source array  
)
```

### Parameters

*src[]*

[in] Reference to an array of source elements to add.

### Return Value

true if successful, false - if you can not add items.

### Example:

```
///--- example for CArrayString::AddArray(const string &[])  
#include <Arrays\ArrayString.mqh>  
///---  
string src[];  
///---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add another array  
    if(!array.AddArray(src))  
    {  
        printf("Array addition error");  
        delete array;  
        return;  
    }  
    ///--- use array  
    ///--- . . .  
    ///--- delete array  
    delete array;  
}
```

## AddArray

Adds to the end of the array elements from another array.

```
bool AddArray(  
    const CArrayString* src      // Pointer to the source  
)
```

### Parameters

*src*

[in] Pointer to an instance of class CArrayString - the source of elements to add.

### Return Value

true if successful, false - if you can not add items.

### Example:

```
///--- example for CArrayString::AddArray(const CArrayString*)  
#include <Arrays\ArrayString.mqh>  
///---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- create source array  
    CArrayString *src=new CArrayString;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    ///--- add source arrays elements  
    ///--- . . .  
    ///--- add another array  
    if(!array.AddArray(src))  
    {  
        printf("Array addition error");  
        delete src;  
        delete array;  
        return;  
    }  
    ///--- delete source array  
    delete src;
```

```
//--- use array  
//--- . . .  
//--- delete array  
delete array;  
}
```

## Insert

Inserts an element in the array to the specified position.

```
bool Insert(  
    string element,    // Element to insert  
    int    pos        // Position  
)
```

### Parameters

*element*

[in] Value of the element to be inserted into an array

*pos*

[in] Position in the array to insert

### Return Value

true if successful, false - if you can not insert the element.

### Example:

```
//--- example for CArrayString::Insert(string,int)  
#include <Arrays\ArrayString.mqh>  
//---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- insert elements  
    for(int i=0;i<100;i++)  
    {  
        if(!array.Insert(IntegerToString(i),0))  
        {  
            printf("Insert error");  
            delete array;  
            return;  
        }  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```

## InsertArray

Inserts an array of elements from another array specified position.

```
bool  InsertArray(  
    const string&  src[],      // Source array  
    int            pos         // Position  
)
```

### Parameters

*src[]*

[in] Reference to an array of source elements to insert

*pos*

[in] Position in the array to insert

### Return Value

true if successful, false - if you can not paste items.

### Example:

```
//--- example for CArrayString::InsertArray(const string &[],int)  
#include <Arrays\ArrayString.mqh>  
//---  
string src[];  
//---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- insert another array  
    if(!array.InsertArray(src,0))  
    {  
        printf("Array inserting error");  
        delete array;  
        return;  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array  
    delete array;  
}
```



## InsertArray

Inserts an array of elements from another array specified position.

```
bool InsertArray(  
    CArrayString* src,      // Pointer to the source  
    int pos               // Position  
)
```

### Parameters

*src*

[in] Pointer to an instance of class CArrayString - the source of elements to insert.

*pos*

[in] Position in the array to insert

### Return Value

true if successful, false - if you can not paste items.

### Example:

```
//--- example for CArrayString::InsertArray(const CArrayString*,int)  
#include <Arrays\ArrayString.mqh>  
//---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source array  
    CArrayString *src=new CArrayString;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    //--- add source arrays elements  
    //--- . . .  
    //--- insert another array  
    if(!array.InsertArray(src,0))  
    {  
        printf("Array inserting error");  
        delete src;  
        delete array;  
    }  
}
```

```
        return;  
    }  
    //-- delete source array  
    delete src;  
    //-- use array  
    //-- . . .  
    //-- delete array  
    delete array;  
}
```

## AssignArray

Copies the array elements from another array.

```
bool AssignArray(  
    const string& src[]      // Source array  
)
```

### Parameters

*src[]*

[in] Reference to an array of source elements to copy.

### Return Value

true if successful, false - if you can not copy the items.

### Example:

```
///--- example for CArrayString::AssignArray(const string &[])  
#include <Arrays\ArrayString.mqh>  
///---  
string src[];  
///---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- assign another array  
    if(!array.AssignArray(src))  
    {  
        printf("Array assigned error");  
        delete array;  
        return;  
    }  
    ///--- use array  
    ///--- . . .  
    ///--- delete array  
    delete array;  
}
```

## AssignArray

Copies the array elements from another array.

```
bool AssignArray(  
    const CArrayString* src      // Pointer to the source  
)
```

### Parameters

*src*

[in] Pointer to an instance of class CArrayString - source of elements to copy.

### Return Value

true if successful, false - if you can not copy the items.

### Example:

```
//--- example for CArrayString::AssignArray(const CArrayString*)  
#include <Arrays\ArrayString.mqh>  
//---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source array  
    CArrayString *src =new CArrayString;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    //--- add source arrays elements  
    //--- . . .  
    //--- assign another array  
    if(!array.AssignArray(src))  
    {  
        printf("Array assigned error");  
        delete src;  
        delete array;  
        return;  
    }  
    //--- arrays is identical  
    //--- delete source array
```

```
delete src;  
//--- use array  
//--- . . .  
//--- delete array  
delete array;  
}
```

## Update

Changes the element at the specified position array.

```
bool Update(  
    int    pos,           // Position  
    string element       // Value  
)
```

### Parameters

*pos*

[in] Position of the element in the array to change

*element*

[in] New value element

### Return Value

true if successful, false - if you can not change the element.

### Example:

```
//--- example for CArrayString::Update(int, string)  
#include <Arrays\ArrayString.mqh>  
//---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- update element  
    if(!array.Update(0,"ABC"))  
    {  
        printf("Update error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

## Shift

Moves an item from a given position in the array to the specified offset.

```
bool Shift(  
    int pos,          // Position  
    int shift         // Shift  
)
```

### Parameters

*pos*

[in] Position of the moved element array

*shift*

[in] The value of displacement (both positive and negative).

### Return Value

true if successful, false - if you can not move the item.

### Example:

```
//--- example for CArrayString::Shift(int,int)  
#include <Arrays\ArrayString.mqh>  
//---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- shift element  
    if(!array.Shift(10,-5))  
    {  
        printf("Shift error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

## Delete

Removes the element from the given position in the array.

```
bool Delete(  
    int pos    // Position  
)
```

### Parameters

*pos*

[in] position removes the element in the array.

### Return Value

true if successful, false - if you can not remove the element.

### Example:

```
///--- example for CArrayString::Delete(int)  
#include <Arrays\ArrayString.mqh>  
///---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    ///--- delete element  
    if(!array.Delete(0))  
    {  
        printf("Delete error");  
        delete array;  
        return;  
    }  
    ///--- delete array  
    delete array;  
}
```



## DeleteRange

Deletes a group of elements from a given position in the array.

```
bool DeleteRange(  
    int from,      // Position of the first element  
    int to         // Position of last element  
)
```

### Parameters

*from*

[in] Position of the first removes the element in the array.

*to*

[in] Position of the last deleted element in the array.

### Return Value

true if successful, false - if you can not remove elements.

### Example:

```
//--- example for CArrayString::DeleteRange(int,int)  
#include <Arrays\ArrayString.mqh>  
//---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- delete elements  
    if(!array.DeleteRange(0,10))  
    {  
        printf("Delete error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

## At

Gets the element from the given position in the array.

```
string At(  
    int pos      // Position  
) const
```

### Parameters

*pos*

[in] Position of the desired element in the array.

### Return Value

The value of the element in case of success, "-" if there was an attempt to get an element of not existing positions (the last error ERR\_OUT\_OF\_RANGE).

### Note

Of course, "" may be a valid value of an array element, so having a value, always check the last error code.

### Example:

```
///--- example for CArrayString::At(int)  
#include <Arrays\ArrayString.mqh>  
///---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    for(int i=0;i<array.Total();i++)  
    {  
        string result=array.At(i);  
        if(result==" " && GetLastError()==ERR_OUT_OF_RANGE)  
        {  
            ///--- Error reading from array  
            printf("Get element error");  
            delete array;  
            return;  
        }  
        ///--- use element  
        ///--- . . .  
    }  
}
```

```
//--- delete array  
delete array;  
}
```

## CompareArray

Compares array with another array.

```
bool CompareArray(  
    const string& src[]      // Source array  
    ) const
```

### Parameters

*src[]*

[in] Reference to an array of source elements for comparison.

### Return Value

true if arrays are equal, false - if not.

### Example:

```
//--- example for CArrayString::CompareArray(const string &[])  
#include <Arrays\ArrayString.mqh>  
//---  
string src[];  
//---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- compare with another array  
    int result=array.CompareArray(src);  
    //--- delete array  
    delete array;  
}
```

## CompareArray

Compares array with another array.

```
bool CompareArrays (
    const CArrayString* src      // Pointer to the source
) const
```

### Parameters

*src*

[in] Pointer to an instance of class CArrayString - the source of elements for comparison.

### Return Value

true if arrays are equal, false - if not.

### Example:

```
///--- example for CArrayString::CompareArray(const CArrayString*)
#include <Arrays\ArrayString.mqh>
///---
void OnStart ()
{
    CArrayString *array=new CArrayString;
    ///---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    ///--- create source array
    CArrayString *src=new CArrayString;
    if(src==NULL)
    {
        printf("Object create error");
        delete array;
        return;
    }
    ///--- add source arrays elements
    ///--- . . .
    ///--- compare with another array
    int result=array.CompareArray(src);
    ///--- delete arrays
    delete src;
    delete array;
}
```

## InsertSort

Inserts element in a sorted array.

```
bool InsertSort(  
    string element    // Element to insert  
)
```

### Parameters

*element*

[in] value of the element to be inserted into a sorted array

### Return Value

true if successful, false - if you can not insert the element.

### Example:

```
///--- example for CArrayString::InsertSort(string)  
#include <Arrays\ArrayString.mqh>  
///---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    ///--- sort array  
    array.Sort();  
    ///--- insert element  
    if(!array.InsertSort("ABC"))  
    {  
        printf("Insert error");  
        delete array;  
        return;  
    }  
    ///--- delete array  
    delete array;  
}
```

## Search

Searches for an element equal to the sample in the sorted array.

```
int Search(  
    string element    // Sample  
) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayString::Search(string)  
#include <Arrays\ArrayString.mqh>  
//---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.Search("ABC")!=-1) printf("Element found");  
    else                        printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchGreat

Searches for an element of more samples in sorted array.

```
int SearchGreat(  
    string element    // Sample  
    ) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayString::SearchGreat(string)  
#include <Arrays\ArrayString.mqh>  
//---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchGreat("ABC")!= -1) printf("Element found");  
    else                             printf("Element not found");  
    //--- delete array  
    delete array;  
}
```



## SearchLess

Searches for an element less than the sample in the sorted array.

```
int SearchLess(  
    string element    // Sample  
    ) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayString:: SearchLess(string)  
#include <Arrays\ArrayString.mqh>  
//---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchLess("ABC")!= -1) printf("Element found");  
    else                             printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchGreatOrEqual

Searches for an element greater than or equal to the sample in the sorted array.

```
int SearchGreatOrEqual(  
    string element    // Sample  
) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayString:: SearchGreatOrEqual(string)  
#include <Arrays\ArrayString.mqh>  
//---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchGreatOrEqual("ABC")!= -1) printf("Element found");  
    else                                     printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchLessOrEqual

Searches for an element less than or equal to the sample in the sorted array.

```
int SearchLessOrEqual(  
    string element    // Sample  
) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
///--- example for CArrayString:: SearchLessOrEqual(string)  
#include <Arrays\ArrayString.mqh>  
///---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    ///--- sort array  
    array.Sort();  
    ///--- search element  
    if(array.SearchLessOrEqual("ABC")!=-1) printf("Element found");  
    else                                printf("Element not found");  
    ///--- delete array  
    delete array;  
}
```

## SearchFirst

Finds the first element equal to the sample in the sorted array.

```
int SearchFirst(  
    string element    // Sample  
    ) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayString:: SearchFirst(string)  
#include <Arrays\ArrayString.mqh>  
//---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchFirst("ABC")!= -1) printf("Element found");  
    else                             printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchLast

Finds the last element equal to the model in sorted array.

```
int SearchLast(  
    string element    // Sample  
    ) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayString:: SearchLast(string)  
#include <Arrays\ArrayString.mqh>  
//---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- search element  
    if(array.SearchLast("ABC")!= -1) printf("Element found");  
    else                             printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchLinear

Searches for the element equal to the sample in the array.

```
int SearchLinear(  
    string element    // Sample  
) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the element was not found.

### Note

The method uses the linear search (or sequential search) algorithm for unsorted arrays.

### Example:

```
///--- example for CArrayString::SearchLinear(string)  
#include <Arrays\ArrayString.mqh>  
///---  
void OnStart()  
{  
    CArrayString *array=new CArrayString;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    ///--- search element  
    if(array.SearchLinear("ABC")!=-1) printf("Element found");  
    else                             printf("Element not found");  
    ///--- delete array  
    delete array;  
}
```

## Save

Saves data array in the file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] Handle to open earlier, with the function FileOpen (...), binary file.

### Return Value

true - if successfully completed, false - if an error.

### Example:

```
//--- example for CArrayString::Save(int)  
#include <Arrays\ArrayString.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CArrayString *array=new CArrayString;  
    //---  
    if(array!=NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add 100 arrays elements  
    for(int i=0;i<100;i++)  
    {  
        array.Add(IntegerToString(i));  
    }  
    //--- open file  
    file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!array.Save(file_handle))  
        {  
            //--- file save error  
            printf("File save: Error %d!",GetLastError());  
            delete array;  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }
```

```
    }  
    delete array;  
}
```



## Load

Loads data array from a file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] Handle to open earlier, with the function FileOpen (...), binary file.

### Return Value

true - if successfully completed, false - if an error.

### Example:

```
//--- example for CArrayString::Load(int)  
#include <Arrays\ArrayString.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CArrayString *array=new CArrayString;  
    //---  
    if(array!=NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- open file  
    file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!array.Load(file_handle))  
        {  
            //--- file load error  
            printf("File load: Error %d!",GetLastError());  
            delete array;  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }  
    //--- use arrays elements  
    for(int i=0;i<array.Total();i++)  
    {  
        printf("Element[%d] = '%s'",i,array.At(i));  
    }  
}
```

```
    }  
    delete array;  
}
```

## Type

Gets the type identifier of the array.

```
virtual int Type() const
```

### Return Value

ID type of the array (for CArrayString - 89).

### Example:

```
//--- example for CArrayString::Type()
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
{
    CArrayString *array=new CArrayString;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- get array type
    int type=array.Type();
    //--- delete array
    delete array;
}
```

## CArrayObj

CArrayObj class is a class of dynamic array of pointers to instances of CObject and his heirs.

### Description

Class CArrayObj provides the ability to work with a dynamic array of pointers to instances of [CObject](#) and his heirs. This gives the possibility to work as a multidimensional dynamic arrays of primitive data types, and the more difficult for organized data structures.

In the class implemented the ability to add / insert / delete elements in an array, sort array, searching in sorted array. In addition, the implemented methods of work with the file.

There are certain [subtleties](#) of the class CArrayObj.

### Declaration

```
class CArrayObj : public CArray
```

### Title

```
#include <Arrays\ArrayObj.mqh>
```

### Class Method

| Attributes                  |                                                                             |
|-----------------------------|-----------------------------------------------------------------------------|
| <a href="#">FreeMode</a>    | Gets the flag memory management                                             |
| <a href="#">FreeMode</a>    | Sets the flag memory management                                             |
| Memory control              |                                                                             |
| <a href="#">Reserve</a>     | Allocates memory to increase the size of the array                          |
| <a href="#">Resize</a>      | Sets a new (smaller) size of the array                                      |
| <a href="#">Shutdown</a>    | Clears the array with a total exemption memory array (not element).         |
| Add methods                 |                                                                             |
| <a href="#">Add</a>         | Adds an element to the end of the array                                     |
| <a href="#">AddArray</a>    | Adds an element to the end of the array                                     |
| <a href="#">Insert</a>      | Inserts an element in the array to the specified position                   |
| <a href="#">InsertArray</a> | Inserts an array of elements from another array with the specified position |
| <a href="#">AssignArray</a> | Copies the array elements from another array                                |
| Update methods              |                                                                             |

|                                    |                                                                             |
|------------------------------------|-----------------------------------------------------------------------------|
| <a href="#">Update</a>             | Changes the element at the specified position array                         |
| <a href="#">Shift</a>              | Moves an item from a given position in the array to the specified offset    |
| <b>Delete methods</b>              |                                                                             |
| <a href="#">Detach</a>             | Gets the element from the specified position and removing it from the array |
| <a href="#">Delete</a>             | Removes the element from the specified position array                       |
| <a href="#">DeleteRange</a>        | Deletes a group of elements from the specified position array               |
| <a href="#">Clear</a>              | Removes all elements of the array without the release of the memory array   |
| <b>Access methods</b>              |                                                                             |
| <a href="#">At</a>                 | Gets the element from the specified position array                          |
| <b>Compare methods</b>             |                                                                             |
| <a href="#">CompareArray</a>       | Compares array with another array                                           |
| <b>Sorted array operations</b>     |                                                                             |
| <a href="#">InsertSort</a>         | Inserts element in a sorted array                                           |
| <a href="#">Search</a>             | Searches for an element equal to the model in sorted array                  |
| <a href="#">SearchGreat</a>        | Searches for an element of more samples in sorted array                     |
| <a href="#">SearchLess</a>         | Searches for an element less than the sample in the sorted array            |
| <a href="#">SearchGreatOrEqual</a> | Searches for an element greater than or equal to the model in sorted array  |
| <a href="#">SearchLessOrEqual</a>  | Searches for an element less than or equal to the model in sorted array     |
| <a href="#">SearchFirst</a>        | Finds the first element equal to the model in sorted array                  |
| <a href="#">SearchLast</a>         | Finds the last element equal to the model in sorted array                   |
| <b>Input/output</b>                |                                                                             |
| <a href="#">Save</a>               | Saves data array in the file                                                |
| <a href="#">Load</a>               | Loads data array from a file                                                |

|      |                                       |
|------|---------------------------------------|
| Type | Gets the type identifier of the array |
|------|---------------------------------------|

Derived classes:

- [CIndicator](#)
- [CIndicators](#)

Practical application of arrays are descendants of class CObject (including all classes of the standard library).

For example, consider the options for two-dimensional array:

```
#include <Arrays\ArrayDouble.mqh>
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
{
    int i,j;
    int first_size=10;
    int second_size=100;
    //--- create array
    CArrayObj *array=new CArrayObj;
    CArrayDouble *sub_array;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- create subarrays
    for(i=0;i<first_size;i++)
    {
        sub_array=new CArrayDouble;
        if(sub_array==NULL)
        {
            delete array;
            printf("Object create error");
            return;
        }
        //--- fill array
        for(j=0;j<second_size;j++)
        {
            sub_array.Add(i*j);
        }
        array.Add(sub_array);
    }
    //--- create array OK
    for(i=0;i<first_size;i++)
    {
```

```

        sub_array=array.At(i);
        for(j=0;j<second_size;j++)
        {
            double element=sub_array.At(j);
            //--- use array element
        }
    }
    delete array;
}

```

## Subtleties

The class has a mechanism to control dynamic memory, so be careful when working with elements of the array.

Mechanism of memory management can be switched on / off using the method `FreeMode (bool)`. By default, the mechanism is enabled.

Accordingly, there are two options for dealing with the class `CArrayObj`:

1. Mechanism of memory management is enabled. (default)

In this case, `CArrayObj` take responsibility for freeing the memory elements after their removal from the array. In this program the user should not free the array elements.

**Example:**

```

int i;
//--- Create an array
CArrayObj *array=new CArrayObj;
//--- Fill array elements
for(i=0;i<10;i++) array.Add(new CObject);
//--- Do something
for(i=0;i<array.Total();i++)
{
    CObject *object=array.At(i);
    //--- Action with an element
    . . .
}
//--- Remove the array with the elements
delete array;

```

2. Mechanism of memory management is turned off.

In this case, `CArrayObj` not otvetstvechaet for freeing the memory elements after their removal from the array. In this program the user must free the array elements.

**Example:**

```

int i;
//--- Create an array

```

```
CArrayObj *array=new CArrayObj;
//--- Disable the mechanism of memory management
array.FreeMode(false);
//--- Fill array elements
for(i=0;i<10;i++) array.Add(new CObject);
//--- Do something
for(i=0;i<array.Total();i++)
{
    CObject *object=array.At(i);
    //--- Action with an element
    . . .
}
//--- Remove array elements
while(array.Total()) delete array.Detach();
//--- Remove empty array
delete array;
```



## FreeMode

Gets the flag memory management.

```
bool FreeMode() const
```

### Return Value

Flag of memory management.

### Example:

```
//--- example for CArrayObj::FreeMode()
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
{
    CArrayObj *array=new CArrayObj;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- get free mode flag
    bool array_free_mode=array.FreeMode();
    //--- delete array
    delete array;
}
```

## FreeMode

Sets the flag memory management.

```
void FreeMode(  
    bool mode    // New flag  
)
```

### Parameters

*mode*

[in] New value of the flag memory management.

### Return Value

None.

### Note

Setting the flag memory management - an important point in the use of class CArrayObj. Since the array elements are pointers to dynamic objects, it is important to determine what to do with them when removing from the array.

If the flag is set, removing an element from the array, the element is automatically deleted by the operator delete. If the flag is not set, it is assumed that a pointer to the deleted object is still somewhere in the user program and will be relieved of it (the program) then.

If the user resets the flag memory management, the user must understand their responsibility for the removal of the array before the completion of the program, because otherwise, is not freed memory occupied by the elements when they create new operator.

When large amounts of data, it could lead, eventually, even to break your terminal. If the user does not reset the flag memory management, there is another "reef".

Using pointers, array, stored somewhere in the local variables, after removing the array will lead to a critical error and crashes the program user. By default, the memory management flag is set, ie the class of the array is responsible for freeing the memory elements.

### Example:

```
//--- example for CArrayObj::FreeMode(bool)  
#include <Arrays\ArrayObj.mqh>  
//---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- reset free mode flag  
    array.FreeMode(false);
```

```
//--- use array  
//--- . . .  
//--- delete array  
delete array;  
}
```

## Reserve

Allocates memory to increase the size of the array.

```
bool Reserve (
    int size      // Number
)
```

### Parameters

*size*

[in] The number of additional elements of the array.

### Return Value

true if successful, false - if there was an attempt to seek the amount is less than or equal to zero, or if the array did not increase.

### Note

To reduce memory fragmentation, increase the size of the array is made with a step previously given through the method of Step (int), or 16 (default).

### Example:

```
///--- example for CArrayObj::Reserve(int)
#include <Arrays\ArrayObj.mqh>
///---
void OnStart()
{
    CArrayObj *array=new CArrayObj;
    ///---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    if(!array.Reserve(1024))
    {
        printf("Reserve error");
        delete array;
        return;
    }
    ///--- use array
    ///--- . . .
    ///--- delete array
    delete array;
}
```

## Resize

Sets a new (smaller) size of the array.

```
bool Resize(  
    int size    // Size  
)
```

### Parameters

*size*

[in] New size of the array.

### Return Value

To reduce fragmentation of memory, change the size of the array is made with a step previously given through the method of Step (int), or 16 (default).

### Note

Changing the size of array allows to use memory in the optimal way. Excess element located to the right are lost. The memory for lost elements is released or not depending on the mode of memory management.

To reduce fragmentation of memory, change the size of the array is made with a step previously given through the method of Step (int), or 16 (default).

### Example:

```
///--- example for CArrayObj::Resize(int)  
#include <Arrays\ArrayObj.mqh>  
///---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    ///--- resize array  
    if(!array.Resize(10))  
    {  
        printf("Resize error");  
        delete array;  
        return;  
    }  
    ///--- delete array  
    delete array;
```

```
}
```

## Clear

Removes all elements of the array without the release of the memory array.

```
void Clear()
```

### Return Value

No.

### Note

If enabled memory management, memory, deleted items are exempt.

### Example:

```
//--- example for CArrayObj::Clear()
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
{
    CArrayObj *array=new CArrayObj;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- add arrays elements
    //--- . . .
    //--- clear array
    array.Clear();
    //--- delete array
    delete array;
}
```

## Shutdown

Clears the array with a total exemption memory array (not element).

```
bool Shutdown()
```

### Return Value

true if successful, false - if an error occurred.

### Note

If enabled memory management, memory, deleted items are exempt.

### Example:

```
//--- example for CArrayObj::Shutdown()
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
{
    CArrayObj *array=new CArrayObj;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- add arrays elements
    //--- . . .
    //--- shutdown array
    if(!array.Shutdown())
    {
        printf("Shutdown error");
        delete array;
        return;
    }
    //--- delete array
    delete array;
}
```



## CreateElement

Creates a new array element at the specified position.

```
bool CreateElement(  
    int index    // Position  
)
```

### Parameters

*index*

[in] position in which you want to create a new element.

### Return Value

true if successful, false - if you can not create element.

### Note

Method CreateElement (int) in class CArrayObj always returns false and does not perform any action. If necessary, in a derived class, method CreateElement (int) should be implemented.

### Example:

```
///--- example for CArrayObj::CreateElement(int)  
#include <Arrays\ArrayObj.mqh>  
///---  
void OnStart()  
{  
    int size=100;  
    CArrayObj *array=new CArrayObj;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- fill array  
    array.Reserve(size);  
    for(int i=0;i<size;i++)  
    {  
        if(!array.CreateElement(i))  
        {  
            printf("Element create error");  
            delete array;  
            return;  
        }  
    }  
    ///--- use array  
    ///--- . . .  
    ///--- delete array  
    delete array;
```

```
}
```

## Add

Adds an element to the end of the array.

```
bool Add(  
    CObject* element    // Element to add  
)
```

### Parameters

*element*

[in] value of the element to add to the array.

### Return Value

true if successful, false - if you can not add an element.

### Note

Element is not added to the array if the value for transmit invalid pointer (such as NULL).

### Example:

```
///--- example for CArrayObj::Add(CObject*)  
#include <Arrays\ArrayObj.mqh>  
///---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add 100 arrays elements  
    for(int i=0;i<100;i++)  
    {  
        if(!array.Add(new CObject))  
        {  
            printf("Element addition error");  
            delete array;  
            return;  
        }  
    }  
    ///--- use array  
    ///--- . . .  
    ///--- delete array  
    delete array;  
}
```

## AddArray

Adds to the end of the array elements from another array.

```
bool AddArray(  
    const CArrayObj * src      // Pointer to the source  
)
```

### Parameters

*src*

[in] Pointer to an instance of class [CArrayDouble](#) - source of elements to add.

### Return Value

true if successful, false - if you can not add items.

### Note

Adding elements of the array in the array is actually copying the pointers. Therefore, when calling the method, there is a "reef" - that there may be a pointer to a dynamic object in more than one variable.

```
//--- example  
extern bool      make_error;  
extern int       error;  
extern CArrayObj *src;  
//--- Create a new instance CArrayObj  
//--- Default memory management is turned on  
CArrayObj *array=new CArrayObj;  
//--- Add (copy) the elements of the array-istochika  
if(array!=NULL)  
    bool result=array.AddArray(src);  
if(make_error)  
{  
    //--- Commit wrongdoing  
    switch(error)  
    {  
        case 0:  
            //--- Remove the array-source, without checking its flag memory management  
            delete src;  
            //--- Result:  
            //--- May appeal  
            //--- For "bitomu" pointer in the array receiver  
            break;  
        case 1:  
            //--- Disable the mechanism of memory management in an array of source  
            if(src.FreeMode()) src.FreeMode(false);  
            //--- But do not remove the array-source  
            //--- Result:  
            //--- After removal of the array, the receiver may appeal
```

```

        //--- For "bitomu" pointer in the array-source
        break;
    case 2:
        //--- Disable the mechanism of memory management in an array of source
        src.FreeMode(false);
        //--- Disable the mechanism of memory management in an array receiver
        array.FreeMode(false);
        //--- Result:
        //--- After the completion of the program, obtain a "memory leak"
        break;
    }
}
else
{
    //--- Disable the mechanism of memory management in an array of source
    if(src.FreeMode()) src.FreeMode(false);
    //--- Remove the array-source
    delete src;
    //--- Result:
    //--- Treatment for an array-receipient be correct
    //--- Delete the array, the receiver will delete its elements
}

```

**Example:**

```

//--- example for CArrayObj::AddArray(const CArrayObj*)
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
{
    CArrayObj *array=new CArrayObj;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- create source array
    CArrayObj *src=new CArrayObj;
    if(src==NULL)
    {
        printf("Object create error");
        delete array;
        return;
    }
    //--- reset free mode flag
    src.FreeMode(false);
    //--- fill source array

```

```
//--- . . .  
//--- add another array  
if(!array.AddArray(src))  
{  
    printf("Array addition error");  
    delete src;  
    delete array;  
    return;  
}  
//--- delete source array without elements  
delete src;  
//--- use array  
//--- . . .  
//--- delete array  
delete array;  
}
```

## Insert

Inserts an element in the array to the specified position.

```
bool Insert(  
    CObject* element,    // Element to insert  
    int      pos         // Position  
)
```

### Parameters

*element*

[in] value of the element to be inserted into an array

*pos*

[in] Position in the array to insert

### Return Value

true if successful, false - if you can not insert the element.

### Note

Element is not added to the array if the value for transmit invalid pointer (such as NULL).

### Example:

```
//--- example for CArrayObj::Insert(CObject*,int)  
#include <Arrays\ArrayObj.mqh>  
//---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- insert elements  
    for(int i=0;i<100;i++)  
    {  
        if(!array.Insert(new CObject,0))  
        {  
            printf("Insert error");  
            delete array;  
            return;  
        }  
    }  
    //--- use array  
    //--- . . .  
    //--- delete array
```

```
delete array;  
}
```



## InsertArray

Inserts an array of elements from another array specified position.

```
bool InsertArray(  
    const CArrayObj* src,      // Pointer to the source  
    int pos                   // Position  
)
```

### Parameters

*src*

[in] Pointer to an instance of class CArrayObj-source elements to insert.

*pos*

[in] Position in the array to insert

### Return Value

true if successful, false - if you can not paste items.

### Note

See: [CArrayObj::AddArray\(const CArrayObj\\*\)](#).

### Example:

```
//--- example for CArrayObj::InsertArray(const CArrayObj*,int)  
#include <Arrays\ArrayObj.mqh>  
//---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source array  
    CArrayObj *src=new CArrayObj;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    //--- reset free mode flag  
    src.FreeMode(false);  
    //--- fill source array  
    //--- . . .  
    //--- insert another array
```

```
if(!array.InsertArray(src,0))
{
    printf("Array inserting error");
    delete src;
    delete array;
    return;
}
//--- delete source array without elements
delete src;
//--- use array
//--- . . .
//--- delete array
delete array;
}
```

## AssignArray

Copies the array elements from another array.

```
bool AssignArray(  
    const CArrayObj* src      // Pointer to the source  
)
```

### Parameters

*src*

[in] Pointer to an instance of class CArrayObj - source of elements to copy.

### Return Value

true if successful, false - if you can not copy the items.

### Note

If the challenge AssignArray array receiver is not empty, all its elements will be removed from the array and, if the flag memory management, memory, deleted items will be released. Array-receiver is an exact copy of the array source. Additionally see [CArrayObj::AddArray\(const CArrayObj\\*\)](#).

### Example:

```
///--- example for CArrayObj::AssignArray(const CArrayObj*)  
#include <Arrays\ArrayObj.mqh>  
///---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- create source array  
    CArrayObj *src=new CArrayObj;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    ///--- reset free mode flag  
    src.FreeMode(false);  
    ///--- fill source array  
    ///--- . . .  
    ///--- assign another array  
    if(!array.AssignArray(src))  
    {
```

```
    printf("Array assigned error");
    delete src;
    delete array;
    return;
}
//--- arrays is identical
//--- delete source array without elements
delete src;
//--- use array
//--- . . .
//--- delete array
delete array;
}
```

## Update

Changes the element at the specified position array.

```
bool Update(  
    int      pos,          // Position  
    CObject* element       // Value  
)
```

### Parameters

*pos*

[in] Position of the element in the array to change

*element*

[in] New value element

### Return Value

true if successful, false - if you can not change the element.

### Note

The element does not change if we as a parameter to pass an invalid pointer (ie NULL). If enabled memory management, memory placeholder released.

### Example:

```
//--- example for CArrayObj::Update(int,CObject*)  
#include <Arrays\ArrayObj.mqh>  
//---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- update element  
    if(!array.Update(0,new CObject))  
    {  
        printf("Update error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

## Shift

Moves an item from a given position in the array to the specified offset.

```
bool Shift(  
    int pos,          // Position  
    int shift         // Shift  
)
```

### Parameters

*pos*

[in] Position of the moved element array

*shift*

[in] The value of displacement (both positive and negative).

### Return Value

true if successful, false - if you can not move the item.

### Example:

```
//--- example for CArrayObj::Shift(int,int)  
#include <Arrays\ArrayObj.mqh>  
//---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- shift element  
    if(!array.Shift(10,-5))  
    {  
        printf("Shift error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

## Detach

Remove an item from a given position in the array.

```
CObject* Detach(  
    int pos    // Position  
)
```

### Parameters

*pos*

[in] Position of the seized item in the array.

### Return Value

Pointer to the removal of elements in case of success, NULL - if you can not remove the element.

### Note

When removed from the array element is not removed in any state of the flag memory management. Pointer to the array element withdrawn from the ingredients of the release after use.

### Example:

```
//--- example for CArrayObj::Detach(int)  
#include <Arrays\ArrayObj.mqh>  
//---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    CObject *object=array.Detach(0);  
    if(object==NULL)  
    {  
        printf("Detach error");  
        delete array;  
        return;  
    }  
    //--- use element  
    //--- . . .  
    //--- delete element  
    delete object;  
    //--- delete array  
    delete array;  
}
```

## Delete

Removes the element from the given position in the array.

```
bool Delete(  
    int pos    // Position  
)
```

### Parameters

*pos*

[in] position removes the element in the array.

### Return Value

true if successful, false - if you can not remove the element.

### Note

If enabled memory management, memory, deleted items are exempt.

### Example:

```
/*--- example for CArrayObj::Delete(int)  
#include <Arrays\ArrayObj.mqh>  
/*---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    /*---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    /*--- add arrays elements  
    /*--- . . .  
    if(!array.Delete(0))  
    {  
        printf("Delete error");  
        delete array;  
        return;  
    }  
    /*--- delete array  
    delete array;  
}
```



## DeleteRange

Deletes a group of elements from a given position in the array.

```
bool DeleteRange(  
    int from,      // Position of the first element  
    int to         // Position of last element  
)
```

### Parameters

*from*

[in] Position of the first removes the element in the array.

*to*

[in] Position of the last deleted element in the array.

### Return Value

true if successful, false - if you can not remove elements.

### Note

If enabled memory management, memory, deleted items are exempt.

### Example:

```
//--- example for CArrayObj::DeleteRange(int,int)  
#include <Arrays\ArrayObj.mqh>  
//---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- delete elements  
    if(!array.DeleteRange(0,10))  
    {  
        printf("Delete error");  
        delete array;  
        return;  
    }  
    //--- delete array  
    delete array;  
}
```

## At

Gets the element from the given position in the array.

```
CObject* At(  
    int pos      // Position  
)
```

### Parameters

*pos*

[in] Position of the desired element in the array.

### Return Value

The value of the element, if successful, NULL-if there was an attempt to get an element of non-existent position.

### Example:

```
///--- example for CArrayObj::At(int)  
#include <Arrays\ArrayObj.mqh>  
///---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add elements  
    ///--- . . .  
    for(int i=0;i<array.Total();i++)  
    {  
        CObject *result=array.At(i);  
        if(result==NULL)  
        {  
            ///--- Error reading from array  
            printf("Get element error");  
            delete array;  
            return;  
        }  
        ///--- use element  
        ///--- . . .  
    }  
    delete array;  
}
```

## CompareArray

Compares array with another array.

```
bool CompareArray(  
    const CArrayObj* src      // Pointer to the source  
    ) const
```

### Parameters

*src*

[in] Pointer to an instance of class CArrayObj - the source of elements for comparison.

### Return Value

true if arrays are equal, false - if not.

### Example:

```
///--- example for CArrayObj::CompareArray(const CArrayObj*)  
#include <Arrays\ArrayObj.mqh>  
///---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- create source array  
    CArrayObj *src=new CArrayObj;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete array;  
        return;  
    }  
    ///--- fill source array  
    ///--- . . .  
    ///--- compare with another array  
    int result=array.CompareArray(src);  
    ///--- delete arrays  
    delete src;  
    delete array;  
}
```

## InsertSort

Inserts element in a sorted array.

```
bool InsertSort(  
    CObject* element    // Element to insert  
)
```

### Parameters

*element*

[in] Value of the element to be inserted into a sorted array

### Return Value

true if successful, false - if you can not insert the element.

### Note

Element is not added to the array if the value for transmit invalid pointer (such as NULL).

### Example:

```
///--- example for CArrayObj::InsertSort(CObject*)  
#include <Arrays\ArrayObj.mqh>  
///---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    ///--- sort array  
    array.Sort();  
    ///--- insert element  
    if(!array.InsertSort(new CObject))  
    {  
        printf("Insert error");  
        delete array;  
        return;  
    }  
    ///--- delete array  
    delete array;  
}
```

## Search

Searches for an element equal to the sample in the sorted array.

```
int Search(  
    CObject* element    // Sample  
    ) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayObj::Search(CObject*)  
#include <Arrays\ArrayObj.mqh>  
//---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- create sample  
    CObject *sample=new CObject;  
    if(sample==NULL)  
    {  
        printf("Sample create error");  
        delete array;  
        return;  
    }  
    //--- set sample attributes  
    //--- . . .  
    //--- search element  
    if(array.Search(sample)!=-1) printf("Element found");  
    else                        printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchGreat

Searches for an element of more samples in sorted array.

```
int SearchGreat(  
    CObject* element    // Sample  
    ) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayObj::SearchGreat(CObject*)  
#include <Arrays\ArrayObj.mqh>  
//---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- create sample  
    CObject *sample=new CObject;  
    if(sample==NULL)  
    {  
        printf("Sample create error");  
        delete array;  
        return;  
    }  
    //--- set sample attributes  
    //--- . . .  
    //--- search element  
    if(array.SearchGreat(sample)!=-1) printf("Element found");  
    else                             printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchLess

Searches for an element less than the sample in the sorted array.

```
int SearchLess(  
    CObject* element    // Sample  
    ) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayObj:: SearchLess(CObject*)  
#include <Arrays\ArrayObj.mqh>  
//---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- create sample  
    CObject *sample=new CObject;  
    if(sample==NULL)  
    {  
        printf("Sample create error");  
        delete array;  
        return;  
    }  
    //--- set sample attributes  
    //--- . . .  
    //--- search element  
    if(array.SearchLess(sample)!=-1) printf("Element found");  
    else                             printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchGreatOrEqual

Searches for an element greater than or equal to the sample in the sorted array.

```
int SearchGreatOrEqual(  
    CObject* element    // Sample  
    ) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
///--- example for CArrayObj::SearchGreatOrEqual(CObject*)  
#include <Arrays\ArrayObj.mqh>  
///---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    ///--- sort array  
    array.Sort();  
    ///--- create sample  
    CObject *sample=new CObject;  
    if(sample==NULL)  
    {  
        printf("Sample create error");  
        delete array;  
        return;  
    }  
    ///--- set sample attributes  
    ///--- . . .  
    ///--- search element  
    if(array.SearchGreatOrEqual(sample)!=-1) printf("Element found");  
    else                                     printf("Element not found");  
    ///--- delete array  
    delete array;  
}
```



## SearchLessOrEqual

Searches for an element less than or equal to the sample in the sorted array.

```
int SearchLessOrEqual(  
    CObject* element    // Sample  
    ) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayObj:: SearchLessOrEqual(CObject*)  
#include <Arrays\ArrayObj.mqh>  
//---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- create sample  
    CObject *sample=new CObject;  
    if(sample==NULL)  
    {  
        printf("Sample create error");  
        delete array;  
        return;  
    }  
    //--- set sample attributes  
    //--- . . .  
    //--- search element  
    if(array.SearchLessOrEqual(sample)!=-1) printf("Element found");  
    else                                  printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchFirst

Finds the first element equal to the sample in the sorted array.

```
int SearchFirst(  
    CObject* element    // Sample  
    ) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
//--- example for CArrayObj::SearchFirst(CObject*)  
#include <Arrays\ArrayObj.mqh>  
//---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    //---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- sort array  
    array.Sort();  
    //--- create sample  
    CObject *sample=new CObject;  
    if(sample==NULL)  
    {  
        printf("Sample create error");  
        delete array;  
        return;  
    }  
    //--- set sample attributes  
    //--- . . .  
    //--- search element  
    if(array.SearchFirst(sample)!=-1) printf("Element found");  
    else                             printf("Element not found");  
    //--- delete array  
    delete array;  
}
```

## SearchLast

Finds the last element equal to the model in sorted array.

```
int SearchLast(  
    CObject* element    // Sample  
    ) const
```

### Parameters

*element*

[in] The sample element to search in the array.

### Return Value

The position of the found element, if successful, -1 - if the item was not found.

### Example:

```
///--- example for CArrayObj:: SearchLast(CObject*)  
#include <Arrays\ArrayObj.mqh>  
///---  
void OnStart()  
{  
    CArrayObj *array=new CArrayObj;  
    ///---  
    if(array==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add arrays elements  
    ///--- . . .  
    ///--- sort array  
    array.Sort();  
    ///--- create sample  
    CObject *sample=new CObject;  
    if(sample==NULL)  
    {  
        printf("Sample create error");  
        delete array;  
        return;  
    }  
    ///--- set sample attributes  
    ///--- . . .  
    ///--- search element  
    if(array.SearchLast(sample)!=-1) printf("Element found");  
    else                             printf("Element not found");  
    ///--- delete array  
    delete array;  
}
```

## Save

Saves data array in the file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] Handle to previously opened by FileOpen (...) function binary file.

### Return Value

true - if successfully completed, false - if an error.

### Example:

```
//--- example for CArrayObj::Save(int)  
#include <Arrays\ArrayObj.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CArrayObj *array=new CArrayObj;  
    //---  
    if(array!=NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add arrays elements  
    //--- . . .  
    //--- open file  
    file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!array.Save(file_handle))  
        {  
            //--- file save error  
            printf("File save: Error %d!",GetLastError());  
            delete array;  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }  
    delete array;  
}
```

## Load

Loads data array from a file.s

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] Handle to open earlier, with the function FileOpen (...), binary file.

### Return Value

true - if successfully completed, false - if an error.

### Note

When reading from the file array to create each element of the method is called [CArrayObj::CreateElement\(int\)](#).

### Example:

```
//--- example for CArrayObj::Load(int)  
#include <Arrays\ArrayObj.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CArrayObj *array=new CArrayObj;  
    //---  
    if(array!=NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- open file  
    file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!array.Load(file_handle))  
        {  
            //--- file load error  
            printf("File load: Error %d!",GetLastError());  
            delete array;  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }  
}
```

```
//--- use arrays elements  
//--- . . .  
//--- delete array  
delete array;  
}
```

## Type

Gets the type identifier of the array.

```
virtual int Type() const
```

### Return Value

ID type of the array (for CArrayObj - 7778).

### Example:

```
//--- example for CArrayObj::Type()
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
{
    CArrayObj *array=new CArrayObj;
    //---
    if(array==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- get array type
    int type=array.Type();
    //--- delete array
    delete array;
}
```

## CList

CList Class is a class of dynamic list of instances of the class CObject and his heirs.

### Description

Class CList provides the ability to work with a list of instances of [CObject](#) and his heirs. In the class implemented the ability to add / insert / delete items in the list, sort the list, search in sorted list. In addition, the implemented methods of work with the file.

There are some subtleties of working with the class CList. The class has a mechanism to control dynamic memory, so be careful when working with elements of the list.

[Subtleties](#) of the mechanism of memory management similar to those described in CArrayObj.

### Declaration

```
class CList : public CObject
```

### Title

```
#include <Arrays\List.mqh>
```

### Class Methods

| Attributes                    |                                                                                       |
|-------------------------------|---------------------------------------------------------------------------------------|
| <a href="#">FreeMode</a>      | Gets the flag memory management when deleting list items.                             |
| <a href="#">FreeMode</a>      | Sets the flag memory management when deleting items in the list                       |
| <a href="#">Total</a>         | Gets the number of elements in the list                                               |
| <a href="#">IsSorted</a>      | Gets flag sorted list                                                                 |
| <a href="#">SortMode</a>      | Gets the version of the sorting                                                       |
| Create methods                |                                                                                       |
| <a href="#">CreateElement</a> | Creates a new item to the list                                                        |
| Add methods                   |                                                                                       |
| <a href="#">Add</a>           | Adds element to the end of the list                                                   |
| <a href="#">Insert</a>        | Inserts element in the list in the specified position                                 |
| Delete methods                |                                                                                       |
| <a href="#">DetachCurrent</a> | Remove an item from the current position of the list without deleting it "physically" |
| <a href="#">DeleteCurrent</a> | Removes the element from the current position in the list                             |



|                                |                                                             |
|--------------------------------|-------------------------------------------------------------|
| <a href="#">Delete</a>         | Removes the element from the specified position in the list |
| <a href="#">Clear</a>          | Removes all list items                                      |
| <b>Navigation</b>              |                                                             |
| <a href="#">IndexOf</a>        | Gets the index of the list item                             |
| <a href="#">GetNodeAtIndex</a> | Gets an item with the specified index of the list           |
| <a href="#">GetFirstNode</a>   | Gets the first element of the list                          |
| <a href="#">GetPrevNode</a>    | Gets the previous element list                              |
| <a href="#">GetCurrentNode</a> | Gets the current list item                                  |
| <a href="#">GetNextNode</a>    | Gets the next item in the list                              |
| <a href="#">GetLastNode</a>    | Gets the last item                                          |
| <b>Ordering methods</b>        |                                                             |
| <a href="#">Sort</a>           | Sort list                                                   |
| <a href="#">MoveToIndex</a>    | Moves the current item list to the specified position       |
| <a href="#">Exchange</a>       | Changes elements of the list seats                          |
| <b>Compare methods</b>         |                                                             |
| <a href="#">CompareList</a>    | Compares the list with another list                         |
| <b>Search methods</b>          |                                                             |
| <a href="#">Search</a>         | Searches for an element equal to the model in sorted list   |
| <b>Input/output</b>            |                                                             |
| virtual <a href="#">Save</a>   | Saves data in the file list                                 |
| virtual <a href="#">Load</a>   | Loads data from file list                                   |
| virtual <a href="#">Type</a>   | Gets the type identifier list                               |

## FreeMode

Gets the flag memory management when deleting list items.

```
bool FreeMode() const
```

### Return Value

Flag of memory management.

### Example:

```
//--- example for CList::FreeMode()
#include <Arrays\List.mqh>
//---
void OnStart()
{
    CList *list=new CList;
    //---
    if(list==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- get free mode flag
    bool list_free_mode=list.FreeMode();
    //--- delete list
    delete list;
}
```

## FreeMode

Sets the flag memory management when deleting list items.

```
void FreeMode(  
    bool mode    // New value  
)
```

### Parameters

*mode*

[in] New value of the flag memory management.

### Note

Setting the flag memory management - an important point in the use of class CList. Since the elements of the list are pointers to dynamic objects, it is important to determine what to do with them when you delete from the list. If the flag is set, then when you delete an item from the list, the item is automatically deleted by the operator delete. If the flag is not set, it is assumed that a pointer to the deleted object is still somewhere in the user program and will be relieved of it (the program) then.

If the user resets the flag memory management, the user must understand their responsibility for the removal of items in the list before completing the program, because otherwise, is not freed memory occupied by the elements when they create new operator. When large amounts of data, it could lead, eventually, even to break your terminal.

If the user does not reset the flag memory management, there is another "reef". Using pointers-list items that are stored somewhere in the local variables, after removing the list, will lead to a critical error and crashes the program user. By default, the memory management flag is set, ie the class list, is responsible for freeing the memory elements.

### Example:

```
//--- example for CList::FreeMode(bool)  
#include <Arrays\List.mqh>  
//---  
void OnStart()  
{  
    CList *list=new CList;  
    //---  
    if(list==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- reset free mode flag  
    list.FreeMode(false);  
    //--- use list  
    //--- . . .  
    //--- delete list  
    delete list;
```

```
}
```

## Total

Gets the number of elements in the list.

```
int Total() const
```

### Return Value

Number of elements in the list.

### Example:

```
//--- example for CList::Total()
#include <Arrays\List.mqh>
//---
void OnStart()
{
    CList *list=new CList;
    //---
    if(list==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- check total
    int total=list.Total();
    //--- use list
    //--- ...
    //--- delete list
    delete list;
}
```

## IsSorted

Gets the flag sorted list.

```
bool IsSorted(  
    int mode=0      // Sorting mode  
) const
```

### Parameters

*mode=0*

[in] Tested version sorting

### Return Value

Flag of the sorted list. If the list is sorted by the specified option? true, otherwise? false.

### Note

Flag of the sorted list can not be changed directly. Flag set by Sort (int) and resets any methods to add / insert.

### Example:

```
///--- example for CList::IsSorted()  
#include <Arrays\List.mqh>  
///---  
void OnStart()  
{  
    CList *list=new CList;  
    ///---  
    if(list==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- check sorted  
    if(list.IsSorted(0))  
    {  
        ///--- use methods for sorted list  
        ///--- ...  
    }  
    ///--- delete list  
    delete list;  
}
```

## SortMode

Gets the version of the sort.

```
int SortMode() const
```

### Return Value

Option sorting, or -1 if the list is not sorted.

### Example:

```
//--- example for CList::SortMode()
#include <Arrays\List.mqh>
//---
void OnStart()
{
    CList *list=new CList;
    //---
    if(list==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- check sort mode
    int sort_mode=list.SortMode();
    //--- use list
    //--- ...
    //--- delete list
    delete list;
}
```

## CreateElement

Creates a new item to the list.

```
CObject* CreateElement()
```

### Return Value

Pointer to the newly created element, if successful, NULL - if you can not create element.

### Note

Method CreateElement () in class CList always returns NULL and does not perform any actions. If necessary, in a derived class, method CreateElement () should be implemented.

### Example:

```
//--- example for CList::CreateElement(int)
#include <Arrays\List.mqh>
//---
void OnStart()
{
    int    size=100;
    CList *list=new CList;
    //---
    if(list==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- fill list
    for(int i=0;i<size;i++)
    {
        CObject *object=list.CreateElement();
        if(object==NULL)
        {
            printf("Element create error");
            delete list;
            return;
        }
        list.Add(object);
    }
    //--- use list
    //--- . . .
    //--- delete list
    delete list;
}
```



## Add

Adds an element to the end of the list.

```
int Add(  
    CObject* element    // Element to add  
)
```

### Parameters

*element*

[in] Value of the element to add to the list.

### Return Value

If successful, it returns the index of added element, or -1 in the case of error.

### Note

Element is not added to the list, if the parameter does not pass valid pointer (ie NULL).

### Example:

```
///--- example for CList::Add(CObject*)  
#include <Arrays\List.mqh>  
///---  
void OnStart()  
{  
    CList *list=new CList;  
    ///---  
    if(list==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add 100 elements  
    for(int i=0;i<100;i++)  
    {  
        if(list.Add(new CObject)==-1)  
        {  
            printf("Element addition error");  
            delete list;  
            return;  
        }  
    }  
    ///--- use list  
    ///--- . . .  
    ///--- delete list  
    delete list;  
}
```

## Insert

Inserts element in the list in the specified position.

```
int Insert(  
    CObject* element,    // Element to insert  
    int      pos         // Position  
)
```

### Parameters

*element*

[in] value of the element to insert in the list

*pos*

[in] Position in the list to insert

### Return Value

If successful, it returns the index of inserted element, or -1 in the case of error.

### Note

Element is not added to the list, if the parameter does not pass valid pointer (ie NULL).

### Example:

```
//--- example for CList::Insert(CObject*,int)  
#include <Arrays\List.mqh>  
//---  
void OnStart()  
{  
    CList *list=new CList;  
    //---  
    if(list==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- insert 100 elements  
    for(int i=0;i<100;i++)  
    {  
        if(list.Insert(new CObject,0)==-1)  
        {  
            printf("Element insert error");  
            delete list;  
            return;  
        }  
    }  
    //--- use list  
    //--- . . .  
    //--- delete list
```

```
delete list;  
}
```

## DetachCurrent

Extracts an element from the current position without its "physical" deletion.

```
CObject* DetachCurrent()
```

### Return Value

Pointer to the removal of elements in case of success, NULL - if you can not remove the element.

### Note

When removed from the list, the item is not removed in any state of the flag memory management.  
Pointer to withdraw from the list of ingredients of the release element after use.

### Example:

```
//--- example for CList::DetachCurrent()
#include <Arrays\List.mqh>
//---
void OnStart()
{
    CList *list=new CList;
    //---
    if(list==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- add list elements
    //--- . . .
    CObject *object=list.DetachCurrent();
    if(object==NULL)
    {
        printf("Detach error");
        delete list;
        return;
    }
    //--- use element
    //--- . . .
    //--- delete element
    delete object;
    //--- delete list
    delete list;
}
```

## DeleteCurrent

Removes the element from the current position in the list.

```
bool DeleteCurrent()
```

### Return Value

true if successful, false - if you can not remove the element.

### Note

If enabled memory management, memory, removes the element is released.

### Example:

```
//--- example for CList::DeleteCurrent()
#include <Arrays\List.mqh>
//---
void OnStart()
{
    CList *list=new CList;
    //---
    if(list==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- add list elements
    //--- . . .
    if(!list.DeleteCurrent())
    {
        printf("Delete error");
        delete list;
        return;
    }
    //--- delete list
    delete list;
}
```

## Delete

Removes the element from the given position in the list.

```
bool Delete(  
    int pos    // Position  
)
```

### Parameters

*pos*

[in] position removes the element in the list.

### Return Value

true if successful, false - if you can not remove the element.

### Note

If enabled memory management, memory, removes the element is released.

### Example:

```
///--- example for CList::Delete(int)  
#include <Arrays\List.mqh>  
///---  
void OnStart()  
{  
    CList *list=new CList;  
    ///---  
    if(list==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    ///--- add list elements  
    ///--- . . .  
    if(!list.Delete(0))  
    {  
        printf("Delete error");  
        delete list;  
        return;  
    }  
    ///--- delete list  
    delete list;  
}
```

## Clear

Removes all elements of the list.

```
void Clear()
```

### Note

If enabled memory management, memory, deleted items are exempt.

### Example:

```
//--- example for CList::Clear()
#include <Arrays\List.mqh>
//---
void OnStart()
{
    CList *list=new CList;
    //---
    if(list==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- add list elements
    //--- . . .
    //--- clear list
    list.Clear();
    //--- delete list
    delete list;
}
```

## IndexOf

Gets the index of the list item.

```
int IndexOf(  
    CObject* element    // Pointer to the element  
)
```

### Parameters

*element*

[in] Pointer to the list item.

### Return Value

Index item in the list, or -1.

### Example:

```
///--- example for CList::IndexOf(CObject*)  
#include <Arrays\List.mqh>  
///---  
void OnStart()  
{  
    CList *list=new CList;  
    ///---  
    if(list==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    CObject *object=new CObject;  
    if(object==NULL)  
    {  
        printf("Element create error");  
        delete list;  
        return;  
    }  
    if(list.Add(object))  
    {  
        int pos=list.IndexOf(object);  
    }  
    ///--- delete list  
    delete list;  
}
```



## GetNodeAtIndex

Gets an item with the specified index of the list.

```
CObject* GetNodeAtIndex(  
    int pos    // position  
)
```

### Parameters

*pos*

[in] item index in the list.

### Returned value

Pointer to the item in case of success, NULL - if you can not get a pointer.

### Example:

```
//--- example for CList::GetNodeAtIndex(int)  
#include <Arrays\List.mqh>  
//---  
void OnStart()  
{  
    CList *list=new CList;  
    //---  
    if(list==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add list elements  
    //--- . . .  
    CObject *object=list.GetNodeAtIndex(10);  
    if(object==NULL)  
    {  
        printf("Get node error");  
        delete list;  
        return;  
    }  
    //--- use element  
    //--- . . .  
    //--- do not delete element  
    //--- delete list  
    delete list;  
}
```

## GetFirstNode

Gets the first element of the list.

```
CObject* GetFirstNode()
```

### Return Value

Pointer to the first item in case of success, NULL - if you can not get a pointer.

### Example:

```
//--- example for CList::GetFirstNode()
#include <Arrays\List.mqh>
//---
void OnStart()
{
    CList *list=new CList;
    //---
    if(list==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- add list elements
    //--- . . .
    CObject *object=list.GetFirstNode();
    if(object==NULL)
    {
        printf("Get node error");
        delete list;
        return;
    }
    //--- use element
    //--- . . .
    //--- do not delete element
    //--- delete list
    delete list;
}
```

## GetPrevNode

Gets the previous element of the list.

```
CObject* GetPrevNode()
```

### Return Value

Pointer to the previous element, if successful, NULL - if you can not get a pointer.

### Example:

```
//--- example for CList::GetPrevNode()
#include <Arrays\List.mqh>
//---
void OnStart()
{
    CList *list=new CList;
    //---
    if(list==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- add list elements
    //--- . . .
    CObject *object=list.GetPrevNode();
    if(object==NULL)
    {
        printf("Get node error");
        delete list;
        return;
    }
    //--- use element
    //--- . . .
    //--- do not delete element
    //--- delete list
    delete list;
}
```

## GetCurrentNode

Gets the current list item.

```
CObject* GetCurrentNode()
```

### Return Value

Pointer to the current item, if successful, NULL - if you can not get a pointer.

### Example:

```
//--- example for CList::GetCurrentNode()
#include <Arrays\List.mqh>
//---
void OnStart()
{
    CList *list=new CList;
    //---
    if(list==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- add list elements
    //--- . . .
    CObject *object=list.GetCurrentNode();
    if(object==NULL)
    {
        printf("Get node error");
        delete list;
        return;
    }
    //--- use element
    //--- . . .
    //--- do not delete element
    //--- delete list
    delete list;
}
```

## GetNextNode

Gets the next item in the list.

```
CObject* GetNextNode()
```

### Return Value

Pointer to the next item if successful, NULL - if you can not get a pointer.

### Example:

```
//--- example for CList::GetNextNode()
#include <Arrays\List.mqh>
//---
void OnStart()
{
    CList *list=new CList;
    //---
    if(list==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- add list elements
    //--- . . .
    CObject *object=list.GetNextNode();
    if(object==NULL)
    {
        printf("Get node error");
        delete list;
        return;
    }
    //--- use element
    //--- . . .
    //--- do not delete element
    //--- delete list
    delete list;
}
```

## GetLastNode

Gets the last element of the list.

```
CObject* GetLastNode()
```

### Return Value

Pointer to the last element in the case of success, NULL - if you can not get a pointer.

### Example:

```
//--- example for CList::GetLastNode()
#include <Arrays\List.mqh>
//---
void OnStart()
{
    CList *list=new CList;
    //---
    if(list==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- add list elements
    //--- . . .
    CObject *object=list.GetLastNode();
    if(object==NULL)
    {
        printf("Get node error");
        delete list;
        return;
    }
    //--- use element
    //--- . . .
    //--- do not delete element
    //--- delete list
    delete list;
}
```

## Sort

Sorts a list.

```
void Sort(  
    int mode    // Sorting mode  
)
```

### Parameters

*mode*

[in] Sorting mode.

### Return Value

No.

### Note

Sorting the list is always in ascending order.

### Example:

```
//--- example for CList::Sort(int)  
#include <Arrays\List.mqh>  
//---  
void OnStart()  
{  
    CList *list=new CList;  
    //---  
    if(list==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- sorting by mode 0  
    list.Sort(0);  
    //--- use list  
    //--- ...  
    //--- delete list  
    delete list;  
}
```

## MoveToIndex

Moves the current item list to the specified position.

```
bool MoveToIndex(  
    int pos      // Position  
)
```

### Parameters

*pos*

[in] Position in the list to move.

### Return Value

true if successful, false - if you can not move the item.

### Example:

```
//--- example for CList::MoveToIndex(int)  
#include <Arrays\List.mqh>  
//---  
void OnStart()  
{  
    CList *list=new CList;  
    //---  
    if(list==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- move current node to begin  
    list.MoveToIndex(0);  
    //--- use list  
    //--- . . .  
    //--- delete list  
    delete list;  
}
```



## Exchange

Changes elements of the list seats.

```
bool Exchange(  
    CObject* node1,    // List item  
    CObject* node2     // List item  
)
```

### Parameters

*node1*

[in] List item

*node2*

[in] List item

### Return Value

true if successful, false - if you can not change the elements in some places.

### Example:

```
//--- example for CList::Exchange(CObject*,CObject*)  
#include <Arrays\List.mqh>  
//---  
void OnStart()  
{  
    CList *list=new CList;  
    //---  
    if(list==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- exchange  
    list.Exchange(list.GetFirstNode(),list.GetLastNode());  
    //--- use list  
    //--- . . .  
    //--- delete list  
    delete list;  
}
```

## CompareList

Compares the list with another list.

```
bool CompareList(  
    CList* list      // With whom we compare  
)
```

### Parameters

*list*

[in] A pointer to an instance of class CList-source elements for comparison.

### Return Value

true if the lists are equal, false - if not.

### Example:

```
//--- example for CList::CompareList(const CList*)  
#include <Arrays\List.mqh>  
//---  
void OnStart()  
{  
    CList *list=new CList;  
    //---  
    if(list==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- create source list  
    CList *src=new CList;  
    if(src==NULL)  
    {  
        printf("Object create error");  
        delete list;  
        return;  
    }  
    //--- fill source list  
    //--- . . .  
    //--- compare with another list  
    bool result=list.CompareList(src);  
    //--- delete lists  
    delete src;  
    delete list;  
}
```

## Search

Searches for an element equal to the model in the sorted list.

```
CObject* Search(  
    CObject* element    // Sample  
)
```

### Parameters

*element*

[in] Sample cell to search for in the list.

### Return Value

Pointer to the found item if successful, NULL - if the item was not found.

### Example:

```
//--- example for CList::Search(CObject*)  
#include <Arrays\List.mqh>  
//---  
void OnStart()  
{  
    CList *list=new CList;  
    //---  
    if(list==NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add lists elements  
    //--- . . .  
    //--- sort list  
    list.Sort(0);  
    //--- create sample  
    CObject *sample=new CObject;  
    if(sample==NULL)  
    {  
        printf("Sample create error");  
        delete list;  
        return;  
    }  
    //--- set sample attributes  
    //--- . . .  
    //--- search element  
    if(list.Search(sample)!=NULL) printf("Element found");  
    else                          printf("Element not found");  
    //--- delete list  
    delete list;  
}
```

## Save

Saves data in the file list.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] handle of the previously opened using the function FileOpen (...) file.

### Return Value

true - if successfully completed, false - if an error.

### Example:

```
//--- example for CList::Save(int)  
#include <Arrays\List.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CList *list=new CList;  
    //---  
    if(list!=NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- add lists elements  
    //--- . . .  
    //--- open file  
    file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!list.Save(file_handle))  
        {  
            //--- file save error  
            printf("File save: Error %d!",GetLastError());  
            delete list;  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }  
    //--- delete list  
    delete list;
```

```
}
```

## Load

Loads list data from a file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] Handle of the previously open, with the function FileOpen (...), binary

### Return Value

true - if successfully completed, false - if an error.

### Note

When reading from the file list items to create each element of the method is called CList::CreateElement ().

### Example:

```
//--- example for CLoad::Load(int)  
#include <Arrays\List.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CList *list=new CList;  
    //---  
    if(list!=NULL)  
    {  
        printf("Object create error");  
        return;  
    }  
    //--- open file  
    file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!list.Load(file_handle))  
        {  
            //--- file load error  
            printf("File load: Error %d!",GetLastError());  
            delete list;  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }  
}
```

```
//--- use list elements  
//--- . . .  
//--- delete list  
delete list;  
}
```

## Type

Gets the type identifier list.

```
virtual int Type()
```

### Return Value

Type identifier list (for CList - 7779).

### Example:

```
//--- example for CList::Type()
#include <Arrays\List.mqh>
//---
void OnStart()
{
    CList *list=new CList;
    //---
    if(list==NULL)
    {
        printf("Object create error");
        return;
    }
    //--- get list type
    int type=list.Type();
    //--- delete list
    delete list;
}
```



## CTreeNode

Class CTreeNode is a class of node of the binary tree CTree.

### Description

Class CTreeNode provides the ability to work with nodes of the binary tree [CTree](#). Options of navigation through the tree is implemented in the class. Besides that methods of work with a file are implemented.

### Declaration

```
class CTreeNode : public CObject
```

### Title

```
#include <Arrays\TreeNode.mqh>
```

### Class Methods

| Attributes                     |                                                      |
|--------------------------------|------------------------------------------------------|
| <a href="#">Owner</a>          | Gets/sets the pointer of the owner node              |
| <a href="#">Left</a>           | Gets/sets the pointer of the left node               |
| <a href="#">Right</a>          | Gets/sets the pointer of the right node              |
| <a href="#">Balance</a>        | Gets the node balance                                |
| <a href="#">BalanceL</a>       | Gets the balance of the left sub-branch of the node  |
| <a href="#">BalanceR</a>       | Gets the balance of the right sub-branch of the node |
| Creation of a new element      |                                                      |
| <a href="#">CreateSample</a>   | Creates a new node instance                          |
| Comparison                     |                                                      |
| <a href="#">RefreshBalance</a> | Recalculates the node balance                        |
| Search                         |                                                      |
| <a href="#">GetNext</a>        | Gets the pointer of the next node                    |
| Input/Output                   |                                                      |
| <a href="#">SaveNode</a>       | Saves the node data to a file                        |
| <a href="#">LoadNode</a>       | Downloads the node data from a file                  |
| virtual <a href="#">Type</a>   | Gets the identifier of the node type                 |

Derived classes:

- [CTree](#)

Trees of CTreeNode class descendants get practical application.

A descendant of class CTreeNode must have predefined methods: [CreateSample](#) that creates a new instance of the descendant class of CTreeNode, [Compare](#) that compares values of key fields of the descendant class of CTreeNode, [Type](#) (if it's necessary to identify a node), [SaveNode](#) and [LoadNode](#) (if it's necessary to work with a file).

Let's consider an example of a CTree descendant class.

```
//+-----+
//|                                     MyTreeNode.mq5 |
//|                                     Copyright 2010, MetaQuotes Software Corp. |
//|                                     http://www.metaquotes.net/ |
//+-----+
#property copyright "2010, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
//---
#include <Arrays\TreeNode.mqh>
//+-----+
//| Describe class derived from CTreeNode. |
//+-----+
//| Class CMyTreeNode. |
//| Purpose: Class of element of a binary tree. |
//|           Descendant of class CTreeNode. |
//+-----+
class CMyTreeNode : public CTreeNode
{
protected:
    //--- user's data
    long      m_long;           // key field of type long
    double    m_double;        // custom variable of type double
    string     m_string;       // custom variable of type string
    datetime  m_datetime;      // custom variable of type datetime

public:
    CMyTreeNode();

    //--- methods of accessing these user's data
    long      GetLong(void)     { return(m_long); }
    void      SetLong(long value) { m_long=value; }
    double    GetDouble(void)   { return(m_double); }
    void      SetDouble(double value) { m_double=value; }
    string     GetString(void)  { return(m_string); }
    void      SetString(string value) { m_string=value; }
    datetime  GetDateTime(void) { return(m_datetime); }
    void      SetDateTime(datetime value) { m_datetime=value; }

    //--- methods of working with files
    virtual bool Save(int file_handle);
    virtual bool Load(int file_handle);
}
```

```

protected:
    virtual int      Compare(const CObject *node,int mode);
    //--- methods of creating class instances
    virtual CTreeNode* CreateSample();
};
//+-----+
//| CMyTreeNode class constructor. |
//| INPUT:  none. |
//| OUTPUT: none. |
//| REMARK: none. |
//+-----+
void CMyTreeNode::CMyTreeNode()
{
    //--- initialization of user's data
    m_long      =0;
    m_double     =0.0;
    m_string     ="";
    m_datetime   =0;
}
//+-----+
//| Comparison with another three node by the specified algorithm. |
//| INPUT:  node - array element to compare, |
//|         mode - identifier of comparison algorithm. |
//| OUTPUT: result of comparison (>0,0,<0). |
//| REMARK: none. |
//+-----+
int CMyTreeNode::Compare(const CObject *node,int mode)
{
    //--- parameter mode is ignored, because tree construction algorithm is the only one
    int res=0;
    //--- explicit type casting
    CMyTreeNode *n=node;
    res=(int) (m_long-n.m_long);
    //---
    return(res);
}
//+-----+
//| Creation of a new class instance. |
//| INPUT:  none. |
//| OUTPUT: pointer to a new instance of class CMyTreeNode. |
//| REMARK: none. |
//+-----+
CTreeNode* CMyTreeNode::CreateSample()
{
    CMyTreeNode *result=new CMyTreeNode;
    //---
    return(result);
}
//+-----+

```

```

//| Write tree node data to a file. |
//| INPUT:  file_handle -handle of a file pre-opened for writing. |
//| OUTPUT: true if OK, otherwise false. |
//| REMARK: none. |
//+-----+
bool CMyTreeNode::Save(int file_handle)
{
    uint i=0,len;
//--- checks
    if(file_handle<0) return(false);
//--- writing user data
//--- writing custom variable of type long
    if(FileWriteLong(file_handle,m_long)!=sizeof(long)) return(false);
//--- writing custom variable of type double
    if(FileWriteDouble(file_handle,m_double)!=sizeof(double)) return(false);
//--- writing custom variable of type string
    len=StringLen(m_string);
//--- write string length
    if(FileWriteInteger(file_handle,len,INT_VALUE)!=INT_VALUE) return(false);
//--- write the string
    if(len!=0 && FileWriteString(file_handle,m_string,len)!=len) return(false);
//--- writing custom variable of type datetime
    if(FileWriteLong(file_handle,m_datetime)!=sizeof(long)) return(false);
//---
    return(true);
}
//+-----+
//| Read tree node data from a file. |
//| INPUT:  file_handle -handle of a file pre-opened for reading. |
//| OUTPUT: true if OK, otherwise false. |
//| REMARK: none. |
//+-----+
bool CMyTreeNode::Load(int file_handle)
{
    uint i=0,len;
//--- checks
    if(file_handle<0) return(false);
//--- reading
    if(FileIsEnding(file_handle)) return(false);
//--- reading custom variable of type char
//--- reading custom variable of type long
    m_long=FileReadLong(file_handle);
//--- reading custom variable of type double
    m_double=FileReadDouble(file_handle);
//--- reading custom variable of type string
//--- read the string length
    len=FileReadInteger(file_handle,INT_VALUE);
//--- read the string
    if(len!=0) m_string=FileReadString(file_handle,len);

```

```
    else        m_string="";  
    ///--- reading custom variable of type datetime  
    m_datetime=FileReadLong(file_handle);  
    ///---  
    return(true);  
}
```

## Owner

Gets the pointer of the owner node.

```
CTreeNode* Owner()
```

### Return Value

Pointer of the node-owner.

## Owner

Sets the pointer of the owner node.

```
void Owner(  
    CTreeNode* node    // node  
)
```

### Parameters

*node*

[in] New value of the pointer of the owner node.

### Return Value

None.

## Left

Gets the pointer of the left node.

```
CTreeNode* Left()
```

### Return Value

Pointer of the left node.

## Left

Sets the pointer of the left node.

```
void Left(  
    CTreeNode* node    // node  
)
```

### Parameters

*node*

[in] New value of the pointer of the left node.

### Return Value

None.

## Right

Gets the pointer of the right node.

```
CTreeNode* Right()
```

### Return Value

The pointer of the right node.

## Right

Sets the pointer of the right node.

```
void Right(  
    CTreeNode* node    // node  
)
```

### Parameters

*node*

[in] New value of the pointer of the right node.

### Return Value

None.



## Balance

Gets the node balance.

```
int Balance() const
```

### Return Value

Node balance.

## BalanceL

Gets the balance of the left sub-branch of the node.

```
int BalanceL() const
```

### Return Value

Balance of the left sub-branch of the node.

## BalanceR

Gets the balance of the right sub-branch of the node.

```
int BalanceR() const
```

### Return Value

Balance of the right sub-branch of the node.

## CreateSample

Creates a new node sample.

```
virtual CTreeNode* CreateSample()
```

### Return Value

Pointer to the new node sample or NULL.

## RefreshBalance

Recalculates the node balance.

```
int RefreshBalance()
```

### Return Value

Node balance.

## GetNext

Gets the pointer of the next node.

```
CTreeNode* GetNext (  
    CTreeNode* node    // node  
)
```

### Parameters

*node*

[in] Node of the search start.

### Return Value

Pointer of the next node.

## SaveNode

Writes node data to a file.

```
bool SaveNode(  
    int file_handle // handle  
)
```

### Parameters

*file\_handle*

[in] Handle of a binary file that was earlier opened for writing.

### Return Value

true in case of success, otherwise false.

## LoadNode

Reads node data from a file.

```
bool LoadNode(  
    int      file_handle,    // handle  
    CTreeNode* main          // node  
)
```

### Parameters

*file\_handle*

[in] Handle of a binary file that was earlier opened for reading.

*main*

[in] Node for data.

### Return Value

true in case of success, otherwise false.



## Type

Gets the identifier of the node type.

```
virtual int Type() const
```

### Return Value

Identifier of the node type.

## CTree

Class CTree is a class of the binary tree of samples of class CTreeNode and its descendants.

### Description

Class CTree provides the possibility to work with a binary tree of [CTreeNode](#) class samples and its descendants. Options of adding/inserting/deleting of three elements and search in a tree are implemented in the class. Besides that, methods of work with a file are implemented.

Note that mechanism of dynamic memory management is not implemented in class CTree (as distinct from classes [CList](#) and [CArrayObj](#)). All tree nodes are deleted with memory release.

### Declaration

```
class CTree : public CTreeNode
```

### Title

```
#include <Arrays\Tree.mqh>
```

### Class Methods

| Attributes                    |                                         |
|-------------------------------|-----------------------------------------|
| <a href="#">Root</a>          | Gets a root node of the tree            |
| Creation of a new element     |                                         |
| <a href="#">CreateElement</a> | Creates a new node instance             |
| Filling                       |                                         |
| <a href="#">Insert</a>        | Adds a node to a tree                   |
| Deletion                      |                                         |
| <a href="#">Detach</a>        | Detaches a specified node from a tree   |
| <a href="#">Delete</a>        | Deletes a specified node from a tree    |
| <a href="#">Clear</a>         | Deletes all nodes of a tree             |
| Search                        |                                         |
| <a href="#">Find</a>          | Searches for a node in a tree by sample |
| Input/output                  |                                         |
| virtual <a href="#">Save</a>  | Saves all the tree data to a file       |
| virtual <a href="#">Load</a>  | Downloads tree data from a file         |
| virtual <a href="#">Type</a>  | Gets identifier of the tree type        |

Trees of CTreeNode class descendants - descendants of class CTree get practical application.

Descendant of class CTree must have a predefined method [CreateElement](#) that creates a new sample of descendant class [CTreeNode](#).

Let's consider an example of descendant class CTree.

```
//+-----+
//|                                     MyTree.mq5 |
//|                                     Copyright 2010, MetaQuotes Software Corp. |
//|                                     http://www.metaquotes.net/ |
//+-----+

#property copyright "2010, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
//---
#include <Arrays\Tree.mqh>
#include "MyTreeNode.mqh"
//---
input int extCountedNodes = 100;
//+-----+
//| Describe class CMyTree derived from CTree. |
//+-----+
//| Class CMyTree. |
//| Purpose: Construction and navigation of a binary search tree. |
//+-----+
class CMyTree : public CTree
{
public:
    //--- methods of search on the tree by custom data
    CMyTreeNode* FindByLong(long find_long);
    //--- method of creation of the tree element
    virtual CTreeNode *CreateElement();
};
//---
CMyTree MyTree;
//+-----+
//| Creation of a new tree node. |
//| INPUT: none. |
//| OUTPUT: pointer to the new tree node of OK, or NULL. |
//| REMARK: none. |
//+-----+
CTreeNode *CMyTree::CreateElement()
{
    CMyTreeNode *node=new CMyTreeNode;
//---
    return (node);
}
//+-----+
//| Search of element in a list by value m_long. |
//| INPUT: find_long - searched value. |
//| OUTPUT: pointer of a found list element, or NULL. |
```

```

//| REMARK: none. |
//+-----+
CMyTreeNode* CMyTree::FindByLong(long find_long)
{
    CMyTreeNode *res=NULL;
    CMyTreeNode *node;
    //--- create a tree node to pass the search parameter
    node=new CMyTreeNode;
    if(node==NULL) return(NULL);
    node.SetLong(find_long);
    //---
    res=Find(node);
    delete node;
    //---
    return(res);
}
//+-----+
//| script "testing of class CMyTree" |
//+-----+
//--- array for string initialization
string str_array[11]={ "p", "oo", "iii", "uuuu", "yyyyy", "ttttt", "rrrr", "eee", "ww", "q", "999" };
//---
int OnStart() export
{
    int i;
    uint pos;
    int beg_time, end_time;
    CMyTreeNode *node; //--- temporary pointer to the sample of class CMyTreeNode
    //---
    printf("Start test %s.", __FILE__);
    //--- Fill out MyTree with samples of class MyTreeNode in the amount of extCountedNodes
    beg_time=GetTickCount();
    for(i=0; i<extCountedNodes; i++)
    {
        node=MyTree.CreateElement();
        if(node==NULL)
        {
            //--- emergency exit
            printf("%s (%4d): create error", __FILE__, __LINE__);
            return(__LINE__);
        }
        NodeSetData(node, i);
        node.SetLong(i);
        MyTree.Insert(node);
    }
    end_time=GetTickCount();
    printf("Filling time of MyTree is %d ms.", end_time-beg_time);
    //--- Create a temporary tree TmpMyTree.
    CMyTree TmpMyTree;

```

```

//--- Detach 50% of tree elements (all even)
//--- and add them to the temporary tree TmpMyTree.
    beg_time=GetTickCount();
    for(i=0;i<extCountedNodes;i+=2)
    {
        node=MyTree.FindByLong(i);
        if(node!=NULL)
            if(MyTree.Detach(node)) TmpMyTree.Insert(node);
    }
    end_time=GetTickCount();
    printf("Deletion time of %d elements from MyTree is %d ms.",extCountedNodes/2,end_t
//--- Return the detached
    node=TmpMyTree.Root();
    while(node!=NULL)
    {
        if(TmpMyTree.Detach(node)) MyTree.Insert(node);
        node=TmpMyTree.Root();
    }
//--- Check work of method Save(int file_handle);
    int file_handle;
    file_handle=FileOpen("MyTree.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
    if(file_handle>=0)
    {
        if(!MyTree.Save(file_handle))
        {
            //--- error writing to a file
            //--- emergency exit
            printf("%s: Error %d in %d!",__FILE__,GetLastError(),__LINE__);
            //--- close file before leaving!!!
            FileClose(file_handle);
            return(__LINE__);
        }
        FileClose(file_handle);
    }
//--- Check work of method Load(int file_handle);
    file_handle=FileOpen("MyTree.bin",FILE_READ|FILE_BIN|FILE_ANSI);
    if(file_handle>=0)
    {
        if(!TmpMyTree.Load(file_handle))
        {
            //--- error reading from file
            //--- emergency exit
            printf("%s: Error %d in %d!",__FILE__,GetLastError(),__LINE__);
            //--- close file before leaving!!!
            FileClose(file_handle);
            return(__LINE__);
        }
        FileClose(file_handle);
    }
}

```

```

//---
    MyTree.Clear();
    TmpMyTree.Clear();
//---
    printf("End test %s. OK!", __FILE__);
//---
    return(0);
}
//+-----+
//| Function of output of node contents to journal |
//+-----+
void NodeToLog(CMyTreeNode *node)
{
    printf("    %I64d,%f,'%s','%s'",
           node.GetLong(), node.GetDouble(),
           node.GetString(), TimeToString(node.GetDateTime()));
}
//+-----+
//| Function of "filling" of node with random values |
//+-----+
void NodeSetData(CMyTreeNode *node, int mode)
{
    if (mode%2==0)
    {
        node.SetLong(mode*MathRand());
        node.SetDouble(MathPow(2.02,mode)*MathRand());
    }
    else
    {
        node.SetLong(mode*(long)(-1)*MathRand());
        node.SetDouble(-MathPow(2.02,mode)*MathRand());
    }
    node.SetString(str_array[mode%10]);
    node.SetDateTime(10000*mode);
}

```

## Root

Gets the root node of the tree.

```
CTreeNode* Root() const
```

### Return Value

Pointer of the root node of the tree.

## CreateElement

Creates a new instance of the node.

```
virtual CTreeNode* CreateElement()
```

### Return Value

Pointer of the new instance of the node or NULL.



## Insert

Adds a node to a tree.

```
CTreeNode* Insert(  
    CTreeNode* new_node    // node  
)
```

### Parameters

*new\_node*

[in] pointer of a node to insert to a tree.

### Return Value

Pointer of the owner node or NULL.

## Detach

Detaches a specified node from a tree.

```
bool Detach(  
    CTreeNode* node    // node  
)
```

### Parameters

*node*

[in] Node pointer to detach.

### Return Value

true in case of success, otherwise false.

### Note

After detachment the node pointer is not released. The tree is balanced.

## Delete

Deletes a specified node from a tree.

```
bool Delete(  
    CTreeNode* node    // node  
)
```

### Parameters

*node*

[in] Node pointer to delete.

### Return Value

true in case of success, otherwise false.

### Note

After deletion a node pointer is released. The tree is balanced.

## Clear

Deletes all nodes of a tree.

```
void Clear()
```

### Return Value

None.

### Note

After deletion node pointers are released.

## Find

Searches for a node in a tree by sample.

```
CTreeNode* Find(  
    CTreeNode* node    // node  
)
```

### Parameters

*node*

[in] Node that contains data-search sample.

### Return Value

Pointer of the found node or NULL.

## Save

Writes tree data to a file.

```
virtual bool Save(  
    int file_handle    // handle  
)
```

### Parameters

*file\_handle*

[in] Handle of a binary file that was earlier opened for writing.

### Return Value

true in case of success, otherwise false.

## Load

Reads tree data to a file.

```
virtual bool Load(  
    int file_handle    // handle  
)
```

### Parameters

*file\_handle*

[in] Handle of a binary file that was earlier opened for reading.

### Return Value

true in case of success, otherwise false.

## Type

Gets identifier of the tree type.

```
virtual int Type() const
```

### Return Value

Identifier of the tree type.



## Graphic Objects

This section contains the technical details of working with classes of graphical objects and a description of the relevant components of the standard library MQL5.

The use of classes of graphical objects, will save time when creating custom programs (scripts, expert).

Standard library MQL5 (in terms of graphical objects) is placed in the working directory of the terminal in the folder Include \ ChartObjects.

| Class/Group                                                         | Description                    |
|---------------------------------------------------------------------|--------------------------------|
| <a href="#"><u>Base class for graphical object CChartObject</u></a> | Base class of a graphic object |
| <a href="#"><u>Lines</u></a>                                        | Group classes "Lines"          |
| <a href="#"><u>Channels</u></a>                                     | Group classes "Channels"       |
| <a href="#"><u>Gann Tools</u></a>                                   | Group classes "Gann"           |
| <a href="#"><u>Fibonacci Tools</u></a>                              | Group classes "Fibonacci"      |
| <a href="#"><u>Elliott Tools</u></a>                                | Group classes "Elliott"        |
| <a href="#"><u>Shapes</u></a>                                       | Group classes "Shapes"         |
| <a href="#"><u>Arrows</u></a>                                       | Group classes "Arrows"         |
| <a href="#"><u>Controls</u></a>                                     | Group classes "Controls"       |

## CChartObject

CChartObject is a base class of graphic objects of chart type of the Standard MQL5 library.

### Description

Class CChartObject provides the simplified access for all of its descendants to MQL5 API functions.

### Declaration

```
class CChartObject : public CObject
```

### Title

```
#include <ChartObjects\ChartObject.mqh>
```

### Class Methods

| Attributes                  |                                                            |
|-----------------------------|------------------------------------------------------------|
| <a href="#">ChartId</a>     | Gets the ID chart, who owns a graphic                      |
| <a href="#">Window</a>      | Gets the number of windows in which the chart is a graphic |
| <a href="#">Name</a>        | Gets/sets the name of a graphic object                     |
| <a href="#">NumPoints</a>   | Gets the number of anchor points                           |
| <b>Assign</b>               |                                                            |
| <a href="#">Attach</a>      | Binds a graphic chart                                      |
| <a href="#">SetPoint</a>    | Sets the anchor point                                      |
| <b>Delete</b>               |                                                            |
| <a href="#">Delete</a>      | Deletes a graphic chart                                    |
| <a href="#">Detach</a>      | Untie a graphic chart                                      |
| <b>Shift</b>                |                                                            |
| <a href="#">ShiftObject</a> | The relative movement of the object                        |
| <a href="#">ShiftPoint</a>  | The relative movement of the object point                  |
| <b>Object properties</b>    |                                                            |
| <a href="#">Time</a>        | Gets/sets the time coordinates of the object point         |
| <a href="#">Price</a>       | Gets/sets the price coordinate of a point object           |
| <a href="#">Color</a>       | Gets/sets the color of the object                          |
| <a href="#">Style</a>       | Gets/sets the line style object                            |
| <a href="#">Width</a>       | Gets/sets the width of the line object                     |

|                                        |                                                      |
|----------------------------------------|------------------------------------------------------|
| <a href="#">BackGround</a>             | Gets/sets the flag drawing object background         |
| <a href="#">Selected</a>               | Gets/sets the flag sit on an object                  |
| <a href="#">Selectable</a>             | Gets/sets the flag selectable object                 |
| <a href="#">Description</a>            | Gets/sets the text of the object                     |
| <a href="#">Tooltip</a>                | Gets/sets the tooltip of the object                  |
| <a href="#">Timeframes</a>             | Gets/sets the mask of flags visibility of the object |
| <a href="#">Z_Order</a>                | Gets/sets the priority of clicking on a chart        |
| <a href="#">CreateTime</a>             | Gets the time object creation                        |
| <b>Levels properties of the object</b> |                                                      |
| <a href="#">LevelsCount</a>            | Gets/sets the number of levels of object             |
| <a href="#">LevelColor</a>             | Gets/sets the color of the line level                |
| <a href="#">LevelStyle</a>             | Gets/sets the line style level                       |
| <a href="#">LevelWidth</a>             | Gets/sets the width of the line level                |
| <a href="#">LevelValue</a>             | Gets/sets the level                                  |
| <a href="#">LevelDescription</a>       | Gets/sets the text level                             |
| <b>Access to MQL5 API functions</b>    |                                                      |
| <a href="#">GetInteger</a>             | Gets the value of the object properties              |
| <a href="#">SetInteger</a>             | Sets the object properties                           |
| <a href="#">GetDouble</a>              | Gets the value of the object properties              |
| <a href="#">SetDouble</a>              | Sets the object properties                           |
| <a href="#">GetString</a>              | Gets the value of the object properties              |
| <a href="#">SetString</a>              | Sets the object properties                           |
| <b>Input/Output</b>                    |                                                      |
| virtual <a href="#">Save</a>           | Virtual method entry in the file                     |
| virtual <a href="#">Load</a>           | Virtual method of reading from a file                |
| virtual <a href="#">Type</a>           | Virtual method of identification                     |

**Derived classes:**

- [CChartObjectArrow](#)
- [CChartObjectBitmap](#)
- [CChartObjectBmpLabel](#)
- [CChartObjectCycles](#)
- [CChartObjectElliottWave3](#)
- [CChartObjectEllipse](#)

- [CChartObjectFiboArc](#)
- [CChartObjectFiboFan](#)
- [CChartObjectFiboTimes](#)
- [CChartObjectHLine](#)
- [CChartObjectRectangle](#)
- [CChartObjectSubChart](#)
- [CChartObjectText](#)
- [CChartObjectTrend](#)
- [CChartObjectTriangle](#)
- [CChartObjectVLine](#)

## ChartId

Gets the ID chart, who owns a graphic object.

```
long ChartId() const
```

### Return Value

Id chart on which the graphic object. If object not found, it returns -1.

### Example:

```
//--- example for CChartObject::ChartId
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
{
    CChartObject object;
    //--- get chart identifier of chart object
    long chart_id=object.ChartId();
}
```

## Window

Gets the number of windows in which the chart is a graphic object.

```
int Window() const
```

### Return Value

Number of windows in which the chart is a graphic object (0 - main window). If object not found, it returns -1.

### Example:

```
//--- example for CChartObject::Window
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
{
    CChartObject object;
    //--- get window of chart object
    int window=object.Window();
}
```

## Name (Get Method)

Gets the name of the graphic object.

```
string Name() const
```

### Return Value

Name of the graphic object tied to an instance of the class. If object not found, returns NULL.

## Name (Set Method)

Sets the name of the graphic object.

```
bool Name(  
    string name    // new name  
)
```

### Parameters

*name*

[in] The new name of the graphic object.

### Return Value

true if successful, false - if you can not change the name.

### Example:

```
//--- example for CChartObject::Name  
#include <ChartObjects\ChartObject.mqh>  
//---  
void OnStart()  
{  
    CChartObject object;  
    //--- get name of chart object  
    string object_name=object.Name();  
    if(object_name!="MyChartObject")  
    {  
        //--- set name of chart object  
        object.Name("MyChartObject");  
    }  
}
```

## NumPoints

Gets the number of anchor points of a graphic object.

```
int NumPoints() const
```

### Return Value

Number of points linking a graphic object that is bound to an instance of the class. If not bound object, it returns 0.

### Example:

```
//--- example for CChartObject::NumPoints
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
{
    CChartObject object;
    //--- get points count of chart object
    int points=object.NumPoints();
}
```



## Attach

Binds a graphical object to an instance of the class.

```
bool Attach(  
    long    chart_id,      // Chart ID  
    string  name,         // Name of the object  
    int     window,       // Chart window  
    int     points        // Number of points  
)
```

### Parameters

*chart\_id*

[out] Chart identifier.

*name*

[in] Name of the graphic object.

*window*

[in] Chart window number (0 - main window).

*points*

[in] Number of points anchor graphic object.

### Return Value

true - if successful, false - if you can not bind object.

### Example:

```
///--- example for CChartObject::Attach  
#include <ChartObjects\ChartObject.mqh>  
///---  
void OnStart()  
{  
    CChartObject object;  
    ///--- attach chart object  
    if(!object.Attach(CharID(), "MyObject", 0, 2))  
    {  
        printf("Object attach error");  
        return;  
    }  
}
```

## SetPoint

Sets the new coordinates of this anchor point graphic object.

```
bool SetPoint(  
    int      point,           // Point number  
    datetime new_time,       // Time coordinate  
    double   new_price       // Price coordinate  
)
```

### Parameters

*point*

[in] Number anchor point graphic object.

*new\_time*

[in] The new value of the coordinates of this point of time bindings.

*new\_price*

[in] New value coordinates of the price specified anchor point.

### Return Value

true - if successful, false - if you can not change the coordinates of the point.

### Example:

```
///--- example for CChartObject::SetPoint  
#include <ChartObjects\ChartObject.mqh>  
///---  
void OnStart()  
{  
    CChartObject object;  
    double       price;  
    ///---  
    if(object.NumPoints()>0)  
    {  
        ///--- set point of chart object  
        object.SetPoint(0,CurrTime(),price);  
    }  
}
```

## Delete

Removes a graphical object with the attached chart.

```
bool Delete()
```

### Return Value

true - if successful, false - if you can not remove the object.

### Example:

```
//--- example for CChartObject::Delete
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
{
    CChartObject object;
    //--- detach chart object
    if(!object.Delete())
    {
        printf("Object delete error");
        return;
    }
}
```

## Detach

Detaches the graphic object.

```
void Detach()
```

### Return Value

None.

### Example:

```
//--- example for CChartObject::Detach
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
{
    CChartObject object;
    //--- detach chart object
    object.Detach();
}
```

## ShiftObject

Moves a graphic object.

```
bool ShiftObject(  
    datetime d_time,      // Increment of time coordinate  
    double    d_price     // Increment of price coordinate  
)
```

### Parameters

*d\_time*

[in] Increment the coordinates of all points of time bindings

*d\_price*

[in] Increment the coordinates of the prices of all waypoints.

### Return Value

true - if successful, false - if you can not move the object.

### Example:

```
//--- example for CChartObject::ShiftObject  
#include <ChartObjects\ChartObject.mqh>  
//---  
void OnStart()  
{  
    CChartObject object;  
    datetime    d_time;  
    double      d_price;  
    //--- shift chart object  
    object.ShiftObject(d_time,d_price);  
}
```

## ShiftPoint

Moves a specified point anchor graphic.

```
bool ShiftPoint(
    int      point,      // Point number
    datetime d_time,     // Increment of time coordinate
    double   d_price     // Increment of price coordinate
)
```

### Parameters

*point*

[in] Number anchor point graphic object.

*d\_time*

[in] Increment the coordinates of time specified anchor point.

*d\_price*

[in] Increment the coordinates of the price specified anchor point.

### Return Value

true - if successful, false - if you can not move the point.

### Example:

```
///--- example for CChartObject::ShiftPoint
#include <ChartObjects\ChartObject.mqh>
///---
void OnStart()
{
    CChartObject object;
    datetime     d_time;
    double       d_price;
    ///---
    if(object.NumPoints()>0)
    {
        ///--- shift point of chart object
        object.ShiftPoint(0,d_time,d_price);
    }
}
```

## Time (Get Method)

The coordinates of time specified anchor point graphic object.

```
datetime Time(  
    int point      // Point number  
) const
```

### Parameters

*point*

[in] Number anchor point graphic object.

### Return Value

Coordinates of time specified anchor point graphic object that is bound to an instance of the class.  
If not bound object or the object is not this point, it returns 0.

## Time (Set Method)

Sets the coordinate of time specified anchor point graphic object.

```
bool Time(  
    int point,      // Point number  
    datetime new_time // Time  
)
```

### Parameters

*point*

[in] Number anchor point graphic object.

*new\_time*

[in] The new value of the coordinates of this point of time anchor graphic object.

### Return Value

true - if successful, false - if you can not change the coordinate of time.

### Example:

```
///--- example for CChartObject::Time  
#include <ChartObjects\ChartObject.mqh>  
///---  
void OnStart()  
{  
    CChartObject object;  
    ///---  
    for(int i=0;i<object.NumPoints();i++)  
    {  
        ///--- get time of point chart object  
        datetime point_time=object.Time(i);  
        if(point_time==0)
```

```
{  
    //--- set time of point chart object  
    object.Time(i,TimeCurrent());  
}  
}  
}
```



## Price (Get Method)

Gets the coordinate of the price specified anchor point graphic object.

```
double Price (
    int point      // Point number
) const
```

### Parameters

*point*

[in] Number anchor point graphic object.

### Return Value

Coordinate prices specified anchor point graphic object that is bound to an instance of the class. If not bound object or the object is not this point, it returns `EMPTY_VALUE`.

## Price (Set Method)

Sets the coordinate of the price specified anchor point graphic object.

```
bool Price (
    int point,      // Point number
    double new_price // Price
)
```

### Parameters

*point*

[in] Number anchor point graphic object.

*new\_price*

[in] News value coordinates of the price specified anchor point graphic object.

### Return Value

true - if successful, false - if you can not change the coordinate prices.

### Example:

```
///--- example for CChartObject::Price
#include <ChartObjects\ChartObject.mqh>
///---
void OnStart()
{
    CChartObject object;
    double price;
    ///---
    for(int i=0;i<object.NumPoints();i++)
    {
        ///--- get price of point chart object
        double point_price=object.Price(i);
```

```
if(point_price!=price)
{
    //--- set price of point chart object
    object.Price(i,price);
}
}
```

## Color (Get Method)

Gets the line color of the graphic object.

```
color Color() const
```

### Return Value

Color line of graphic object, assigned to the class instance. If there is no object assigned, it returns CLR\_NONE.

## Color (Set Method)

Sets the color of the line for the graphic object.

```
bool Color(  
    color new_color    // New color  
)
```

### Parameters

*new\_color*

[in] New value line color graphic object.

### Return Value

true - if successful, false - if you can not change the color.

### Example:

```
//--- example for CChartObject::Color  
#include <ChartObjects\ChartObject.mqh>  
//---  
void OnStart()  
{  
    CChartObject object;  
    //--- get color of chart object  
    color object_color=object.Color();  
    if(object_color!=clrRed)  
    {  
        //--- set color of chart object  
        object.Color(clrRed);  
    }  
}
```

## Style (Get Method)

Gets the line style graphic.

```
ENUM_LINE_STYLE Style() const
```

### Return Value

Style line of the graphic object, assigned to the class instance. If there is no object assigned, it returns WRONG\_VALUE.

## Style (Set Method)

Sets the line style graphic.

```
bool Style(  
    ENUM_LINE_STYLE new_style    // Style  
)
```

### Parameters

*new\_style*

[in] New value-style line drawing object.

### Return Value

true - if successful, false - if you can not change the style.

### Example:

```
//--- example for CChartObject::Style  
#include <ChartObjects\ChartObject.mqh>  
//---  
void OnStart()  
{  
    CChartObject object;  
    //--- get style of chart object  
    ENUM_LINE_STYLE style=object.Style();  
    if(style!=STYLE_SOLID)  
    {  
        //--- set style of chart object  
        object.Style(STYLE_SOLID);  
    }  
}
```

## Width (Get Method)

Gets the thickness of the line graphic object.

```
int Width() const
```

### Return Value

The thickness of the line graphic object that is bound to an instance of the class. If not bound object, it returns -1.

## Width (Set Method)

Sets the thickness of the line graphic object.

```
bool Width(  
    int new_width    // Thickness  
)
```

### Parameters

*new\_width*

[in] The new value of the thickness of the line graphic object.

### Return Value

true - if successful, false - if you can not change the thickness.

### Example:

```
//--- example for CChartObject::Width  
#include <ChartObjects\ChartObject.mqh>  
//---  
void OnStart()  
{  
    CChartObject object;  
    //--- get width of chart object  
    int width=object.Width();  
    if(width!=1)  
    {  
        //--- set width of chart object  
        object.Width(1);  
    }  
}
```

## Background (Get Method)

Gets the flag drawing a graphic object in the background.

```
bool Background() const
```

### Return Value

Flag of drawing in the background, a graphic object that is bound to an instance of the class. If not bound object returns false.

## Background (Set Method)

Sets the flag drawing a graphic object in the background.

```
bool Background(  
    bool background    // Value of the flag  
)
```

### Parameters

*background*

[in] New value of the flag drawing a graphic object in the background.

### Return Value

true - if successful, false - if you can not change the flag.

### Example:

```
///--- example for CChartObject::Background  
#include <ChartObjects\ChartObject.mqh>  
///---  
void OnStart()  
{  
    CChartObject object;  
    ///--- get background flag of chart object  
    bool background_flag=object.Background();  
    if(!background_flag)  
    {  
        ///--- set background flag of chart object  
        object.Background(true);  
    }  
}
```

## Selected (Get Method)

Gets the "selected" flag of a graphic object. In other words - is a graphic object selected or not.

```
bool Selected() const
```

### Return Value

The "selected" flag of a graphic object that is bound to an instance of the class. If there is no bound object, returns false.

## Selected (Set Method)

Sets the "selected" flag of a graphic object.

```
bool Selected(  
    bool selected    // Value of the flag  
)
```

### Parameters

*selected*

[in] New value of the "selected" flag of a graphic object.

### Return Value

true - if successful, false - if you can not change the flag.

### Example:

```
//--- example for CChartObject::Selected  
#include <ChartObjects\ChartObject.mqh>  
//---  
void OnStart()  
{  
    CChartObject object;  
    //--- get the "selected" flag of chart object  
    bool selected_flag=object.Selected();  
    if(selected_flag)  
    {  
        //--- set the "selected" flag of chart object  
        object.Selected(false);  
    }  
}
```

## Selectable (Get Method)

Gets the "selectable" flag of a graphic object. In other words - can you select a graphic object or not.

```
bool Selectable() const
```

### Return Value

The "selectable" flag of a graphic object that is bound to an instance of the class. If there is no bound object, returns false.

## Selectable (Set Method)

Sets the "selectable" flag of a graphic object.

```
bool Selectable(  
    bool selectable    // Value of the flag  
)
```

### Parameters

*selectable*

[in] New value of the "selectable" flag of a graphic object.

### Return Value

true - if successful, false - if you can not change the flag.

### Example:

```
//--- example for CChartObject::Selectable  
#include <ChartObjects\ChartObject.mqh>  
//---  
void OnStart()  
{  
    CChartObject object;  
    //--- get the "selectable" flag of chart object  
    bool selectable_flag=object.Selectable();  
    if(selectable_flag)  
    {  
        //--- set the "selectable" flag of chart object  
        object.Selectable(false);  
    }  
}
```



## Description (Get Method)

Gets a description (text) graphic object.

```
string Description() const
```

### Return Value

Description (text) graphic object that is bound to an instance of the class. If no bound object, it returns NULL.

## Description (Set Method)

Sets the description (text) graphic object.

```
bool Description(  
    string text    // Text  
)
```

### Parameters

*text*

[in] New value descriptions (text) graphic object.

### Return Value

true - if successful, false - if you can not change the description (text).

### Example:

```
//--- example for CChartObject::Description  
#include <ChartObjects\ChartObject.mqh>  
//---  
void OnStart()  
{  
    CChartObject object;  
    //--- get description of chart object  
    string description=object.Description();  
    if(description=="")  
    {  
        //--- set description of chart object  
        object.Description("MyObject");  
    }  
}
```

## Tooltip (Get Method)

Gets the text of the tooltip of graphic object.

```
string Tooltip() const
```

### Returned value

The text of a tooltip. If the object is not assigned, it returns NULL.

## Tooltip (Set Method)

Sets the text of a tooltip.

```
bool Tooltip(  
    string new_tooltip    // new text of a tooltip  
)
```

### Parameters

*new\_tooltip*

[in] New value of a tooltip

### Returned value

true - if successful, false if tooltip change has failed.

### Note:

If the property is not set, then the tooltip generated automatically by the terminal is shown. A tooltip can be disabled by setting the "\n" (line feed) value.

## Timeframes (Get Method)

Gets the visibility flag graphic object.

```
int Timeframes() const
```

### Return Value

Flags visibility graphic object that is bound to an instance of the class. If not bound object, it returns 0.

## Timeframes (Set Method)

Sets the visibility flag graphic.

```
bool Timeframes (
    int new_timeframes    // Visibility flags
)
```

### Parameters

*new\_timeframes*

[in] New flags visibility graphic object.

### Return Value

true - if successful, false - if you can not change the flags of visibility.

### Example:

```
///--- example for CChartObject::Timeframes
#include <ChartObjects\ChartObject.mqh>
///---
void OnStart()
{
    CChartObject object;
    ///--- get timeframes of chart object
    int timeframes=object.Timeframes();
    if(!(timeframes&OBJ_PERIOD_H1))
    {
        ///--- set timeframes of chart object
        object.Timeframes(timeframes|OBJ_PERIOD_H1);
    }
}
```

## Z\_Order (Get Method)

Gets the value of priority of graphical object of clicking on a chart ([CHARTEVENT\\_CLICK](#)).

```
long Z_Order() const
```

### Return Value

Priority of a graphic object, assigned to the class instance. If there is no object assigned, it returns 0.

## Z\_Order (Set Method)

Sets priority of graphical object of clicking on a chart ([CHARTEVENT\\_CLICK](#)).

```
bool Z_Order(  
    long value    // new priority  
)
```

### Parameters

*value*

[in] New value of priority of a graphical object of clicking on a chart ([CHARTEVENT\\_CLICK](#)).

### Return Value

true - if successful, false - if you can not change the priority.

### Note

Z\_Order is a priority of a graphical object for receiving events of clicking on a chart ([CHARTEVENT\\_CLICK](#)). By setting the value, greater than 0(default value) you can increase the object priority.

## CreateTime

Gets the time to create graphical object.

```
datetime CreateTime() const
```

### Return Value

Time to create graphical object that is bound to an instance of the class. If not bound object, it returns 0.

### Example:

```
//--- example for CChartObject::CreateTime
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
{
    CChartObject object;
    //--- get create time of chart object
    datetime create_time=object.CreateTime();
}
```

## LevelsCount (Get Method)

Gets the number of levels of graphical object.

```
int LevelsCount() const
```

### Return Value

Number of levels of graphical object that is bound to an instance of the class. If not bound object, it returns 0.

## LevelsCount (Set Method)

Sets the number of levels of graphical object.

```
bool LevelsCount (
    int levels      // Number of levels
)
```

### Parameters

*levels*

[in] the number of new levels of graphic object.

### Return Value

true - if successful, false - if you can not change the number of levels.

### Example:

```
///--- example for CChartObject::LevelsCount
#include <ChartObjects\ChartObject.mqh>
///---
void OnStart()
{
    CChartObject object;
    ///--- get levels count of chart object
    int levels_count=object.LevelsCount();
    ///--- set levels count of chart object
    object.LevelsCount(levels_count+1);
}
```

## LevelColor (Get Method)

Gets the line color specified level of graphic object.

```
color LevelColor(  
    int level    // Level number  
) const
```

### Parameters

*level*

[in] Number of levels of graphical object.

### Return Value

Color Line level specified graphic object that is bound to an instance of the class. If not bound object or the object is no specified level, it returns CLR\_NONE.

## LevelColor (Set Method)

Sets the line color specified level of graphic object.

```
bool LevelColor(  
    int level,        // Level number  
    color new_color    // New color  
)
```

### Parameters

*level*

[in] Number of levels of graphical object.

*new\_color*

[in] New value line color of the level of graphic object.

### Return Value

true - if successful, false - if you can not change the color.

### Example:

```
///--- example for CChartObject::LevelColor  
#include <ChartObjects\ChartObject.mqh>  
///---  
void OnStart()  
{  
    CChartObject object;  
    ///---  
    for(int i=0;i<object.LevelsCount();i++)  
    {  
        ///--- get level color of chart object  
        color level_color=object.LevelColor(i);  
        if(level_color!=clrRed)
```

```
{  
    //--- set level color of chart object  
    object.LevelColor(i,clrRed);  
}  
}  
}
```



## LevelStyle (Get Method)

Gets the line style specified level of graphical object.

```
ENUM_LINE_STYLE LevelStyle(  
    int level      // Level number  
) const
```

### Parameters

*level*

[in] Number of levels of graphical object.

### Return Value

Line style specified level graphical object that is bound to an instance of the class. If not bound object or the object is no specified level of returns WRONG\_VALUE.

## LevelStyle (Set Method)

Sets the line style specified level of graphical object.

```
int LevelStyle(  
    int level,      // Level number  
    ENUM_LINE_STYLE style // Line Style  
)
```

### Parameters

*level*

[in] Number of levels of graphical object.

*style*

[in] New value-style line level specified graphical object.

### Return Value

true - if successful, false - if you can not change the style.

### Example:

```
//--- example for CChartObject::LevelStyle  
#include <ChartObjects\ChartObject.mqh>  
//---  
void OnStart()  
{  
    CChartObject object;  
    //---  
    for(int i=0;i<object.LevelsCount();i++)  
    {  
        //--- get level style of chart object  
        ENUM_LINE_STYLE level_style=object.LevelStyle(i);  
        if(level_style!=STYLE_SOLID)
```

```
{  
    //--- set level style of chart object  
    object.LevelStyle(i,STYLE_SOLID);  
}  
}  
}
```

## LevelWidth (Get Method)

Gets the line thickness specified level of graphic object.

```
int LevelWidth(  
    int level      // Level number  
) const
```

### Parameters

*level*

[in] Number of levels of graphical object.

### Return Value

The thickness of the lines of the level of graphical object that is bound to an instance of the class. If not bound object or the object is no specified level, it returns -1.

## LevelWidth (Set Method)

Finds the last element equal to the model in sorted array.

```
bool LevelWidth(  
    int level,      // Level number  
    int new_width   // New width  
)
```

### Parameters

*level*

[in] Number of levels of graphical object.

*new\_width*

[in] New value line thickness specified level graphical object.

### Return Value

position of the found element, if successful, -1 - if the item was not found.

### Example:

```
///--- example for CChartObject::LevelWidth  
#include <ChartObjects\ChartObject.mqh>  
///---  
void OnStart()  
{  
    CChartObject object;  
    ///---  
    for(int i=0;i<object.LevelsCount();i++)  
    {  
        ///--- get level width of chart object  
        int level_width=object.LevelWidth(i);  
        if(level_width!=1)
```

```
{
    //--- set level width of chart object
    object.LevelWidth(i,1);
}
}
```

## LevelValue (Get Method)

Gets the value of the level of graphic object.

```
double LevelValue(  
    int level      // Level number  
) const
```

### Parameters

*level*

[in] Number of levels of graphical object.

### Return Value

The value of this level of graphical object that is bound to an instance of the class. If not bound object or the object is no level specified, returns EMPTY\_VALUE.

## LevelValue (Set Method)

Sets the value of the specified level of graphic object.

```
bool LevelValue(  
    int level,      // Level number  
    double new_value // New value  
)
```

### Parameters

*level*

[in] Number of levels of graphical object.

*new\_value*

[in] New value of the level of graphic object.

### Return Value

true - if successful, false - if you can not change the level.

### Example:

```
//--- example for CChartObject::LevelValue  
#include <ChartObjects\ChartObject.mqh>  
//---  
void OnStart()  
{  
    CChartObject object;  
    //---  
    for(int i=0;i<object.LevelsCount();i++)  
    {  
        //--- get level value of chart object  
        double level_value=object.LevelValue(i);  
        if(level_value!=0.1*i)
```

```
{  
    //--- set level value of chart object  
    object.LevelValue(i,0.1*i);  
}  
}  
}
```

## LevelDescription (Get Method)

Gets a description (text) of the level of graphical object.

```
string LevelDescription(  
    int level // Level number  
) const
```

### Parameters

*level*

[in] Number of levels of graphical object.

### Return Value

Description (text) of the level of graphical object that is bound to an instance of the class. If not bound object or the object is no specified level, returns NULL.

## LevelDescription (Set Method)

Sets the description (text) of the level of graphical object.

```
bool LevelDescription(  
    int level, // Level number  
    string text // Text  
)
```

### Parameters

*level*

[in] Number of level of graphical object.

*text*

[in] New value of description (text) of the level of graphic object.

### Return Value

true - if successful, false - if you can not change the description (text).

### Example:

```
///--- example for CChartObject::LevelDescription  
#include <ChartObjects\ChartObject.mqh>  
///---  
void OnStart()  
{  
    CChartObject object;  
    ///---  
    for(int i=0;i<object.LevelsCount();i++)  
    {  
        ///--- get level description of chart object  
        string level_description=object.LevelDescription(i);  
        if(level_description=="")
```

```
{
    ///--- set level description of chart object
    object.LevelDescription(i,"Level_"+IntegerToString(i));
}
}
```



## GetInteger

Provides simplified access to the functions of API MQL5 [ObjectGetInteger\(\)](#) for integer-value properties (of type bool, char, uchar, short, ushort, int, uint, ling, ulong, datetime, color) bound to an instance of the class graphic. There are two versions of a function call:

### Getting a property value without checking the correctness

```
long GetInteger (
    ENUM_OBJECT_PROPERTY_INTEGER prop_id,      // Identifier of integer-property
    int modifier=-1                          // Modifier
) const
```

#### Parameters

*prop\_id*

[in] ID of double-graphic properties.

*modifier=-1*

[in] Modifier (index) double-features.

#### Return Value

If successful, it returns the value of integer-type property, if error, it returns 0.

### Getting a property value in verifying the correctness of such treatment

```
bool GetInteger (
    ENUM_OBJECT_PROPERTY_INTEGER prop_id,      // Identifier of integer-property
    int modifier,                             // Modifier
    long& value                                // Link to variable
) const
```

#### Parameters

*prop\_id*

[in] ID integer-graphic properties of the object.

*modifier*

[in] Modifier (index) integer-property.

*value*

[out] Reference to a variable to accommodate the integer-value properties.

#### Return Value

true - if successful, false - if you can not get integer-property.

#### Example:

```
//--- example for CChartObject::GetInteger
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart ()
```

```
{
    CChartObject object;
    //--- get color of chart object by easy method
    printf("Objects color is %s",ColorToString(object.GetInteger(OBJPROP_COLOR),true));
    //--- get color of chart object by classic method
    long color_value;
    if(!object.GetInteger(OBJPROP_COLOR,0,color_value))
    {
        printf("Get integer property error %d",GetLastError());
        return;
    }
    else
        printf("Objects color is %s",color_value);
    for(int i=0;i<object.LevelsCount();i++)
    {
        //--- get levels width by easy method
        printf("Level %d width is %d",i,object.GetInteger(OBJPROP_LEVELWIDTH,i));
        //--- get levels width by classic method
        long width_value;
        if(!object.GetInteger(OBJPROP_LEVELWIDTH,i,width_value))
        {
            printf("Get integer property error %d",GetLastError());
            return;
        }
        else
            printf("Level %d width is %d",i,width_value);
    }
}
```

## SetInteger

Provides simplified access to the functions of API MQL5 [ObjectSetInteger\(\)](#) to change the integer-properties (with types bool, char, uchar, short, ushort, int, uint, ling, ulong, datetime, color) bound to an instance of the class graphic. There are two versions of a function call:

### Setting a property value that does not require a modifier

```
bool SetInteger(  
    ENUM_OBJECT_PROPERTY_INTEGER prop_id,    // Identifier of integer-property  
    long value                                // Value  
)
```

#### Parameters

*prop\_id*

[in] ID integer-graphic properties of the object.

*value*

[in] new mutable integer-value properties.

### Setting a property value indicating the modifier

```
bool SetInteger(  
    ENUM_OBJECT_PROPERTY_INTEGER prop_id,    // Identifier of integer-property  
    int modifier,                            // Modifier  
    long value                                // Value  
)
```

#### Parameters

*prop\_id*

[in] ID integer-graphic properties of the object.

*modifier*

[in] Modifier (index) integer-property.

*value*

[in] new mutable integer-value properties.

#### Return Value

true - if successful, false - if you can not change the integer-property.

#### Example:

```
//--- example for CChartObject::SetInteger  
#include <ChartObjects\ChartObject.mqh>  
//---  
void OnStart()  
{  
    CChartObject object;  
    //--- set new color of chart object
```

```
if(!object.SetInteger(OBJPROP_COLOR,clrRed))
{
    printf("Set integer property error %d",GetLastError());
    return;
}
for(int i=0;i<object.LevelsCount();i++)
{
    //--- set levels width
    if(!object.SetInteger(OBJPROP_LEVELWIDTH,i,i))
    {
        printf("Set integer property error %d",GetLastError());
        return;
    }
}
}
```

## GetDouble

Provides simplified access to the functions of API MQL5 [ObjectGetDouble\(\)](#) to get the values double-properties (having type float and double) of the graphic object, assigned to the class instance. There are two versions of a function call:

### Getting a property value without checking the correctness

```
double  GetDouble (
    ENUM_OBJECT_PROPERTY_DOUBLE prop_id,      // Identifier of double-property
    int      modifier=-1                      // Modifier
) const
```

#### Parameters

*prop\_id*  
[in] ID of double-graphic properties.

*modifier=-1*  
[in] Modifier (index) double-features.

#### Return Value

If successful, it returns the value of property of double type, if error, it returns EMPTY\_VALUE.

### Getting a property value in verifying the correctness of such treatment

```
bool  GetDouble (
    ENUM_OBJECT_PROPERTY_DOUBLE prop_id,      // Identifier of double-property
    int      modifier,                      // Modifier
    double&   value                        // Link to variable
) const
```

#### Parameters

*prop\_id*  
[in] ID of double-graphic properties.

*modifier*  
[in] Modifier (index) double-features.

*value*  
[out] Reference to a variable to accommodate the double-value properties.

#### Return Value

true - if successful, false - if you can not get a double-feature.

#### Example:

```
//--- example for CChartObject::GetDouble
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart ()
```

```
{
    CChartObject object;
    ///---
    for(int i=0;i<object.LevelsCount();i++)
    {
        ///--- get levels value by easy method
        printf("Level %d value=%f",i,object.GetDouble(OBJPROP_LEVELVALUE,i));
        ///--- get levels value by classic method
        double value;
        if(!object.SetDouble(OBJPROP_LEVELVALUE,i,value))
        {
            printf("Get double property error %d",GetLastError());
            return;
        }
        else
            printf("Level %d value=%f",i,value);
    }
}
```

## SetDouble

Provides simplified access to the functions of API MQL5 [ObjectSetDouble\(\)](#) to change the double-properties (having type float and double) bound to an instance of the class graphic object. There are two versions of a function call:

### Setting a property value that does not require a modifier

```
bool SetDouble (
    ENUM_OBJECT_PROPERTY_DOUBLE prop_id,    // Identifier of double-property
    double value                             // Value
)
```

#### Parameters

*prop\_id*

[in] ID of double-graphic properties.

*value*

[in] New value mutable double-features.

### Setting a property value indicating the modifier

```
bool SetDouble (
    ENUM_OBJECT_PROPERTY_DOUBLE prop_id,    // Identifier of double-property
    int modifier,                          // Modifier
    double value                             // Value
)
```

#### Parameters

*prop\_id*

[in] ID of double-graphic properties.

*modifier*

[in] Modifier (index) of double-property.

*value*

[in] New value mutable double-property.

#### Return Value

true - if successful, false - if you can not change the double-feature.

#### Example:

```
//--- example for CChartObject::SetDouble
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart ()
{
    CChartObject object;
//---
```

```
for(int i=0;i<object.LevelsCount();i++)
{
    //--- set level value of chart object
    if(!object.SetDouble(OBJPROP_LEVELVALUE,i,0.1*i))
    {
        printf("Set double property error %d",GetLastError());
        return;
    }
}
```



## GetString

Provides simplified access to the functions of API MQL5 [ObjectGetString\(\)](#) for string-value properties, is bound to an instance of the class `graphic object`. There are two versions of a function call:

### Getting a property value without checking the correctness

```
string GetString(
    ENUM_OBJECT_PROPERTY_STRING prop_id,      // Identifier of string-property
    int modifier=-1                          // Modifier
) const
```

#### Parameters

*prop\_id*  
[in] Identifier string-properties of graphic object.

*modifier=-1*  
[in] Modifier (index) string-properties.

#### Return Value

Value of string-property.

### Getting a property value in verifying the correctness of such treatment

```
bool GetString(
    ENUM_OBJECT_PROPERTY_STRING prop_id,      // Identifier of string-property
    int modifier,                            // Modifier
    string& value                             // Link to variable
) const
```

#### Parameters

*prop\_id*  
[in] Identifier string-properties of graphic object.

*modifier*  
[in] Modifier (index) string-properties.

*value*  
[out] Reference to a variable to accommodate the string-value properties.

#### Return Value

true - if successful, false - if you can not get a string-property.

#### Example:

```
//--- example for CChartObject::GetString
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
{
```

```
CChartObject object;
string      value;
//--- get name of chart object by easy method
printf("Object name is '%s'",object.GetString(OBJPROP_NAME));
//--- get name of chart object by classic method
if(!object.GetString(OBJPROP_NAME,0,value))
{
    printf("Get string property error %d",GetLastError());
    return;
}
else
    printf("Object name is '%s'",value);
for(int i=0;i<object.LevelsCount();i++)
{
    //--- get levels description by easy method
    printf("Level %d description is '%s'",i,object.GetString(OBJPROP_LEVELTEXT,i));
    //--- get levels description by classic method
    if(!object.GetString(OBJPROP_LEVELTEXT,i,value))
    {
        printf("Get string property error %d",GetLastError());
        return;
    }
    else
        printf("Level %d description is '%s'",i,value);
}
}
```

## SetString

Provides simplified access to the functions of API MQL5 [ObjectSetString\(\)](#) to change the properties of string-tied to the instance of the class graphic object. There are two versions of a function call:

### Setting a property value that does not require a modifier

```
bool SetString(  
    ENUM_OBJECT_PROPERTY_STRING prop_id,    // Identifier of string-property  
    string value                            // Value  
)
```

#### Parameters

*prop\_id*

[in] Identifier string-properties of graphic object.

*value*

[in] New value mutable string-properties.

### Setting a property value indicating the modifier

```
bool SetString(  
    ENUM_OBJECT_PROPERTY_STRING prop_id,    // Identifier of string-property  
    int modifier,                          // Modifier  
    string value                            // Value  
)
```

#### Parameters

*prop\_id*

[in] Identifier string-properties of graphic object.

*modifier*

[in] Modifier (index) string-properties.

*value*

[in] New value mutable string-properties.

#### Return Value

true - if successful, false - if you can not change the string-property.

#### Example:

```
///--- example for CChartObject::SetString  
#include <ChartObjects\ChartObject.mqh>  
///---  
void OnStart()  
{  
    CChartObject object;  
    ///--- set new name of chart object  
    if(!object.SetString(OBJPROP_NAME, "MyObject"))
```

```
{
    printf("Set string property error %d", GetLastError());
    return;
}
for(int i=0; i<object.LevelsCount(); i++)
{
    ///--- set levels description
    if(!object.SetString(OBJPROP_LEVELTEXT, i, "Level_" + IntegerToString(i)))
    {
        printf("Set string property error %d", GetLastError());
        return;
    }
}
}
```

## Save

Saves parameters of the object in the file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] handle of the previously opened using the function FileOpen (...) file.

### Return Value

true - if successfully completed, false - if an error.

### Example:

```
//--- example for CChartObject::Save  
#include <ChartObjects\ChartObject.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CChartObject object=new CChartObject;  
    //--- set object parameters  
    //--- . . .  
    //--- open file  
    file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!object.Save(file_handle))  
        {  
            //--- file save error  
            printf("File save: Error %d!",GetLastError());  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }  
}
```

## Load

Loads the parameters of the object from the file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] handle of the previously opened using the function FileOpen (...) file.

### Return Value

true - if successfully completed, false - if an error.

### Example:

```
//--- example for CChartObject::Load  
#include <ChartObjects\ChartObject.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CChartObject object;  
    //--- open file  
    file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!object.Load(file_handle))  
        {  
            //--- file load error  
            printf("File load: Error %d!",GetLastError());  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }  
    //--- use object  
    //--- . . .  
}
```

## Type

Gets the type identifier graphic object.

```
virtual int Type() const
```

### Return Value

Object type identifier (0x8888 for [CChartObject](#)).

### Example:

```
//--- example for CChartObject::Type
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
{
    CChartObject object;
    //--- get object type
    int type=object.Type();
}
```

## Objects Lines

A group of graphic objects "Lines".

This section contains the technical details of working with a group of classes of graphical objects "Lines" and a description of the relevant components of the MQL5 Standard Library .

| Class name                               | Object                               |
|------------------------------------------|--------------------------------------|
| <a href="#">CChartObjectVLine</a>        | Graphic object "Vertical Line"       |
| <a href="#">CChartObjectHLine</a>        | Graphic object "Horizontal Line"     |
| <a href="#">CChartObjectTrend</a>        | Graphic object "Trend Line"          |
| <a href="#">CChartObjectTrendByAngle</a> | Graphic object "Trend Line by Angle" |
| <a href="#">CChartObjectCycles</a>       | Graphic object "Cyclic Lines"        |

See also

[Object types](#), [Graphic objects](#)



# CChartObjectVLine

Class CChartObjectVLine is a class for simplified access to "Vertical Line" graphic object properties.

## Description

Class CChartObjectVLine provides access to "Vertical Line" object properties.

## Declaration

```
class CChartObjectVLine : public CChartObject
```

## Title

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

## Class Methods

|                              |                                        |
|------------------------------|----------------------------------------|
| Create                       |                                        |
| <a href="#">Create</a>       | Creates graphic object "Vertical Line" |
| Input/output                 |                                        |
| virtual <a href="#">Type</a> | Virtual method of identification       |

See also

[Object types](#), [Graphic objects](#)

## Create

Creates graphic object "Vertical Line".

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string    name,          // Object name  
    int       window,        // Chart window  
    datetime  time           // Time coordinate  
)
```

### Parameters

*chart\_id*

[in] Chart identifier (0 - current chart).

*name*

[in] An unique name of the object to create.

*window*

[in] Chart window number (0 - base window).

*time*

[in] Time coordinate of the anchor point.

### Returned value

true if successful, false if error.

## Type

Returns object type identifier of the graphic object.

```
int Type() const
```

### Returned value

Object type identifier (OBJ\_VLINE for [CChartObjectVLine](#)).

# CChartObjectHLine

Class CChartObjectHLine is a class for simplified access to "Horizontal Line" graphic object properties.

## Description

Class CChartObjectHLine provides access to "Horizontal Line" object properties.

## Declaration

```
class CChartObjectHLine : public CChartObject
```

## Title

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

## Class Methods

| Create                       |                                          |
|------------------------------|------------------------------------------|
| <a href="#">Create</a>       | Creates graphic object "Horizontal Line" |
| Input/output                 |                                          |
| virtual <a href="#">Type</a> | Virtual method of identification         |

### See also

[Object types](#), [Graphic objects](#)

## Create

Creates graphic object "Horizontal Line".

```
bool Create(  
    long    chart_id,    // Chart identifier  
    string  name,        // Object name  
    long    window,      // Chart window  
    double  price        // Price coordinate  
)
```

### Parameters

*chart\_id*

[in] Chart identifier (0 - current chart).

*name*

[in] An unique name of the object to create.

*window*

[in] Chart window number (0 - base window).

*price*

[in] Price coordinate of the anchor point.

### Returned value

true if successful, false if error.

## Type

Returns object type identifier of the graphic object.

```
int Type() const
```

### Returned value

Object type identifier (OBJ\_HLINE for [CChartObjectHLine](#)).

## CChartObjectTrend

Class CChartObjectTrend is a class for simplified access to "Trend Line" graphic object properties.

### Description

Class CChartObjectTrend provides access to "Trend Line" object properties.

### Declaration

```
class CChartObjectTrend : public CChartObject
```

### Title

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

### Class Methods

| Create                       |                                      |
|------------------------------|--------------------------------------|
| <a href="#">Create</a>       | Creates graphic object "Trend Line"  |
| Properties                   |                                      |
| <a href="#">RayLeft</a>      | Gets/Sets property "Ray Left"        |
| <a href="#">RayRight</a>     | Gets/Sets property "Ray Right"       |
| Input/output                 |                                      |
| virtual <a href="#">Save</a> | Virtual method for writing to file   |
| virtual <a href="#">Load</a> | Virtual method for reading from file |
| virtual <a href="#">Type</a> | Virtual method of identification     |

#### Derived classes:

- [CChartObjectChannel](#)
- [CChartObjectFibo](#)
- [CChartObjectFiboChannel](#)
- [CChartObjectFiboExpansion](#)
- [CChartObjectGannFan](#)
- [CChartObjectGannGrid](#)
- [CChartObjectPitchfork](#)
- [CChartObjectRegression](#)
- [CChartObjectStdDevChannel](#)
- [CChartObjectTrendByAngle](#)

#### See also

[Object types](#), [Graphic objects](#)

## Create

Creates graphic object "Trend Line".

```
bool Create(  
    long      chart_id,    // Chart identifier  
    string    name,        // Object name  
    int       window,      // Chart window  
    datetime  time1,       // 1st time coordinate  
    double    price1,      // 1st price coordinate  
    datetime  time2,       // 2nd time coordinate  
    double    price2       // 2nd price coordinate  
)
```

### Parameters

*chart\_id*

[in] Chart identifier (0 - current chart).

*name*

[in] An unique name of the object to create.

*window*

[in] Chart window number (0 - base window).

*time1*

[in] Time coordinate of the first anchor point.

*price1*

[in] Price coordinate of the first anchor point.

*time2*

[in] Time coordinate of the second anchor point.

*price2*

[in] Price coordinate of the second anchor point.

### Returned value

true if successful, false if error.



## RayLeft (Get Method)

Gets the value of "Ray Left" property.

```
bool RayLeft () const
```

### Returned value

The value of "Ray Left" property, assigned to the class instance. If there is no object assigned, it returns false.

## RayLeft (Set Method)

Sets new flag value for the "Ray Left" property.

```
bool RayLeft (  
    bool ray      // flag  
)
```

### Parameters

*ray*

[in] New value of the "Ray Left" property.

### Returned value

true if successful, false if flag hasn't changed error.

## RayRight (Get Method)

Gets the value of "Ray Right" property.

```
bool RayRight() const
```

### Returned value

The value of "Ray Right" property, assigned to the class instance. If there is no object assigned, it returns false.

## RayRight (Set Method)

Sets new flag value for the "Ray Right" property.

```
bool RayRight (  
    bool ray      // flag  
)
```

### Parameters

*ray*

[in] New value of the "Ray Right" property.

### Returned value

true if successful, false if flag hasn't changed error.

## Save

Saves object parameters to file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] handle of the file already opened using by function FileOpen(...).

### Returned value

true if successful, false if error.

## Load

Loads object parameters from file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] handle of the file already opened using by function FileOpen(...).

### Returned value

true if successful, false if error.

## Type

Returns object type identifier of the graphic object.

```
int Type() const
```

### Returned value

Object type identifier (OBJ\_TREND for [CChartObjectTrend](#)).

## CChartObjectTrendByAngle

Class CChartObjectTrendByAngle is a class for simplified access to "Trend Line by Angle" graphic object properties.

### Description

Class CChartObjectTrendByAngle provides access to "Trend Line by Angle" object properties.

### Declaration

```
class CChartObjectTrendByAngle : public CChartObjectTrend
```

### Title

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

### Class Methods

| Create                       |                                              |
|------------------------------|----------------------------------------------|
| <a href="#">Create</a>       | Creates graphic object "Trend Line by Angle" |
| Properties                   |                                              |
| <a href="#">Angle</a>        | Gets/Sets property "Angle"                   |
| Input/output                 |                                              |
| virtual <a href="#">Type</a> | Virtual method of identification             |

Derived classes:

- [CChartObjectGannLine](#)

See also

[Object types](#), [Graphic objects](#)

## Create

Creates graphic object "Trend Line by Angle"

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string    name,          // Object name  
    long      window,        // Chart window  
    datetime  time1,         // 1st time coordinate  
    double    price1,        // 1st price coordinate  
    datetime  time2,         // 2nd time coordinate  
    double    price2         // 2nd price coordinate  
)
```

### Parameters

*chart\_id*

[in] Chart identifier (0 - current chart).

*name*

[in] An unique name of the object to create.

*window*

[in] Chart window number (0 - base window).

*time1*

[in] Time coordinate of the first anchor point.

*price1*

[in] Price coordinate of the first anchor point.

*time2*

[in] Time coordinate of the second anchor point.

*price2*

[in] Price coordinate of the second anchor point.

### Returned value

true if successful, false if error.

## Angle (Get Method)

Gets the value of "Angle" property.

```
double Angle() const
```

### Returned value

The value of "Angle" property, assigned to the class instance. If there is no object assigned, it returns EMPTY\_VALUE.

## Angle (Set Method)

Sets new value for the "Angle" property.

```
bool Angle(  
    double angle    // Angle  
)
```

### Parameters

*angle*

[in] New value of the "Angle" property.

### Returned value

true if successful, false if property hasn't changed.



## Type

Returns object type identifier of the graphic object.

```
int Type() const
```

### Returned value

Object type identifier (OBJ\_TRENDBYANGLE for [CChartObjectTrendByAngle](#)).

# CChartObjectCycles

Class CChartObjectCycles is a class for simplified access to "Cyclic Lines" graphic object properties.

## Description

Class CChartObjectCycles provides access to "Cyclic Lines" object properties.

## Declaration

```
class CChartObjectCycles : public CChartObject
```

## Title

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

## Class Methods

|                              |                                      |
|------------------------------|--------------------------------------|
| Create                       |                                      |
| <a href="#">Create</a>       | Creates graphic object "Cycle Lines" |
| Input/output                 |                                      |
| virtual <a href="#">Type</a> | Virtual method of identification     |

See also

[Object types](#), [Graphic objects](#)

## Create

Creates graphic object "Cyclic Lines"

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string     name,         // Object name  
    long      window,        // Chart window  
    datetime   time1,        // 1st time coordinate  
    double     price1,       // 1st price coordinate  
    datetime   time2,        // 2nd time coordinate  
    double     price2        // 2nd price coordinate  
)
```

### Parameters

*chart\_id*

[in] Chart identifier (0 - current chart).

*name*

[in] An unique name of the object to create.

*window*

[in] Chart window number (0 - base window).

*time1*

[in] Time coordinate of the first anchor point.

*price1*

[in] Price coordinate of the first anchor point.

*time2*

[in] Time coordinate of the second anchor point.

*price2*

[in] Price coordinate of the second anchor point.

### Returned value

true if successful, false if error.

## Type

Returns object type identifier of the graphic object.

```
int Type() const
```

### Returned value

Object type identifier (OBJ\_CYCLES for [CChartObjectCycles](#)).

## Objects Channels

A group of graphic objects "Channels".

This section contains the technical details of working with a group of classes of graphical objects "Channels" and a description of the relevant components of the MQL5 Standard Library .

| Class name                                       | Object                                       |
|--------------------------------------------------|----------------------------------------------|
| <a href="#"><u>CChartObjectChannel</u></a>       | Graphic object "Equidistant Channel"         |
| <a href="#"><u>CChartObjectRegression</u></a>    | Graphic object "Linear Regression Channel"   |
| <a href="#"><u>CChartObjectStdDevChannel</u></a> | Graphic object "Standard deviations Channel" |
| <a href="#"><u>CChartObjectPitchfork</u></a>     | Graphic object "Andrew's Pitchfork"          |

See also

[Object types](#), [Graphic objects](#)

# CChartObjectChannel

The CChartObjectChannel is a class for simplified access to "Equidistant Channel" graphic object properties.

## Description

The class CChartObjectChannel provides access to "Equidistant Channel" object properties.

## Declaration

```
class CChartObjectChannel : public CChartObjectTrend
```

## Title

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

## Class Methods

| Create                       |                                              |
|------------------------------|----------------------------------------------|
| <a href="#">Create</a>       | Creates graphic object "Equidistant Channel" |
| Input/output                 |                                              |
| virtual <a href="#">Type</a> | Virtual method of identification             |

### See also

[Object types](#), [Graphic objects](#)

## Create

Creates graphic object "Equidistant Channel"

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string    name,          // Object name  
    int       window,        // Chart window  
    datetime  time1,         // Time coordinate  
    double    price1,  
    datetime  time2,  
    double    price2,  
    datetime  time3,  
    double    price3  
)
```

### Parameters

*chart\_id*

[in] Chart ID (0 - current chart).

*name*

[in] An unique name of the object to create.

*window*

[in] Chart window number (0 - base window).

*time1*

[in] Time coordinate of the first anchor point.

*price1*

[in] Price coordinate of the first anchor point.

*time2*

[in] Time coordinate of the second anchor point.

*price2*

[in] Price coordinate of the second anchor point.

*time3*

[in] Time coordinate of the third anchor point.

*price3*

[in] Price coordinate of the third anchor point.

### Returned value

true if successful, false if error.

## Type

Returns object type identifier of the graphic object.

```
int Type() const
```

### Returned value

Object type identifier (OBJ\_CHANNEL for [CChartObjectChannel](#)).



# CChartObjectRegression

Class CChartObjectRegression is a class for simplified access to "Linear Regression Channel" graphic object properties.

## Description

Class CChartObjectRegression provides access to "Linear Regression Channel" object properties.

## Declaration

```
class CChartObjectRegression : public CChartObjectTrend
```

## Title

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

## Class Methods

| Create              |                                                    |
|---------------------|----------------------------------------------------|
| <u>Create</u>       | Creates graphic object "Linear Regression Channel" |
| Input/output        |                                                    |
| virtual <u>Type</u> | Virtual method of identification                   |

## See also

[Object types](#), [Graphic objects](#)

## Create

Creates graphic object "Linear Regression Channel"

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string     name,         // Object name  
    long      window,        // Chart window  
    datetime   time1,        // First time coordinate  
    datetime   time2         // Second time coordinate  
)
```

### Parameters

*chart\_id*

[in] Chart identifier (0 - current chart).

*name*

[in] An unique name of the object to create.

*window*

[in] Chart window number (0 - base window).

*time1*

[in] Time coordinate of the first anchor point.

*time2*

[in] Time coordinate of the second anchor point.

### Returned value

true if successful, false if error.

## Type

Returns object type identifier of the graphic object.

```
int Type() const
```

### Returned value

Object type identifier (OBJ\_REGRESSION for [CChartObjectRegression](#)).

# CChartObjectStdDevChannel

Class CChartObjectStdDevChannel is a class for simplified access to "Standard Deviation Channel" graphic object properties.

## Description

Class CChartObjectStdDevChannel provides access to "Standard Deviation Channel" object properties.

## Declaration

```
class CChartObjectStdDevChannel : public CChartObjectTrend
```

## Title

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

## Class Methods

| Create                       |                                                     |
|------------------------------|-----------------------------------------------------|
| <a href="#">Create</a>       | Creates graphic object "Standard Deviation Channel" |
| Properties                   |                                                     |
| <a href="#">Deviations</a>   | Gets/Sets property "Deviation"                      |
| Input/output                 |                                                     |
| virtual <a href="#">Save</a> | Virtual method for writing to file                  |
| virtual <a href="#">Load</a> | Virtual method for reading from file                |
| virtual <a href="#">Type</a> | Virtual method of identification                    |

### See also

[Object types](#), [Graphic objects](#)

## Create

Creates graphic object "Standard Deviation Channel"

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string    name,          // Object name  
    int       window,        // Chart window  
    datetime  time1,         // First time coordinate  
    datetime  time2,         // Second time coordinate  
    double    deviation      // Deviation  
)
```

### Parameters

*chart\_id*

[in] Chart identifier (0 - current chart).

*name*

[in] An unique name of the object to create.

*window*

[in] Chart window number (0 - base window).

*time1*

[in] Time coordinate of the first anchor point.

*time2*

[in] Time coordinate of the second anchor point.

*deviation*

[in] Numerical value for "Deviation" property.

### Returned value

true if successful, false if error.

## Deviation (Get Method)

Gets numerical value for "Deviation" property.

```
double Deviation() const
```

### Returned value

Numerical value of "Deviation" property, assigned to the class instance. If there is no object assigned, it returns EMPTY\_VALUE.

## Deviation (Set Method)

Sets numerical value for "Deviation" property.

```
bool Deviation(  
    double deviation    // Deviation  
)
```

### Parameters

*deviation*

[in] New value for "Deviation" property.

### Returned value

true if successful, false if property hasn't changed.

## Save

Saves object parameters to file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] handle of the file already opened using by function FileOpen(...) .

### Returned value

true if successful, false if error.

## Load

Loads object parameters from file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] handle of the file already opened using by function FileOpen(...) .

### Returned value

true if successful, false if error.



## Type

Returns object type identifier of the graphic object.

```
int Type() const
```

### Returned value

Object type identifier (OBJ\_STDDEVCHANNEL for [CChartObjectStdDevChannel](#)).

# CChartObjectPitchfork

Class CChartObjectPitchfork is a class for simplified access to "Andrew's Pitchfork" graphic object properties.

## Description

Class CChartObjectPitchfork provides access to "Andrew's Pitchfork" object properties.

## Declaration

```
class CChartObjectPitchfork : public CChartObjectTrend
```

## Title

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

## Class Methods

| Create                       |                                             |
|------------------------------|---------------------------------------------|
| <a href="#">Create</a>       | Creates graphic object "Andrew's Pitchfork" |
| Input/output                 |                                             |
| virtual <a href="#">Type</a> | Virtual method of identification            |

### See also

[Object types](#), [Graphic objects](#)

## Create

Creates graphic object "Andrew's Pitchfork"

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string     name,         // Object name  
    long      window,        // Chart window  
    datetime   time1,        // 1st time coordinate  
    double     price1,       // 1st price coordinate  
    datetime   time2,        // ...  
    double     price2,  
    datetime   time3,  
    double     price3  
)
```

### Parameters

*chart\_id*

[in] Chart identifier (0 - current chart).

*name*

[in] An unique name of the object to create.

*window*

[in] Chart window number (0 - base window).

*time1*

[in] Time coordinate of the first anchor point.

*price1*

[in] Price coordinate of the first anchor point..

*time2*

[in] Time coordinate of the second anchor point.

*price2*

[in] Price coordinate of the first anchor point.

*time3*

[in] Time coordinate of the third anchor point.

*price3*

[in] Price coordinate of the third anchor point.

### Returned value

true if successful, false if error.

## Type

Returns object type identifier of the graphic object.

```
int Type() const
```

### Returned value

Object type identifier (OBJ\_PITCHFORK for [CChartObjectPitchfork](#)).

## Gann Tools

A group of graphic objects "Gann Tools".

This section contains the technical details of working with a group of classes of graphical objects "Gann Tools" and a description of the relevant components of the MQL5 Standard Library .

| Class name                                  | Object                     |
|---------------------------------------------|----------------------------|
| <a href="#"><u>CChartObjectGannLine</u></a> | Graphic object "Gann Line" |
| <a href="#"><u>CChartObjectGannFan</u></a>  | Graphic object "Gann Fan"  |
| <a href="#"><u>CChartObjectGannGrid</u></a> | Graphic object "Gann Grid" |

See also

[Object types](#), [Graphic objects](#)

## CChartObjectGannLine

Class CChartObjectGannLine is a class for simplified access to "Gann Line" graphic object properties.

### Description

Class CChartObjectGannLine provides access to "Gann Line" object properties.

### Declaration

```
class CChartObjectGannLine : public CChartObjectTrendByAngle
```

### Title

```
#include <ChartObjects\ChartObjectsGann.mqh>
```

### Class Methods

| Create                       |                                      |
|------------------------------|--------------------------------------|
| <a href="#">Create</a>       | Creates graphic object "Gann Line"   |
| Properties                   |                                      |
| <a href="#">PipsPerBar</a>   | Gets/Sets property "Scale"           |
| Input/output                 |                                      |
| virtual <a href="#">Save</a> | Virtual method for writing to file   |
| virtual <a href="#">Load</a> | Virtual method for reading from file |
| virtual <a href="#">Type</a> | Virtual method of identification     |

### See also

[Object types](#), [Graphic objects](#)

## Create

Creates graphic object "Gann Line".

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string     name,         // Object name  
    int        window,       // Chart window  
    datetime   time1,        // First time coordinate  
    double     price1,       // First price coordinate  
    datetime   time2,        // Second time coordinate  
    double     ppb           // Pips per bar  
)
```

### Parameters

*chart\_id*

[in] Chart identifier (0 - current chart).

*name*

[in] An unique name of the object to create.

*window*

[in] Chart window number (0 - base window).

*time1*

[in] Time coordinate of the first anchor point.

*price1*

[in] Price coordinate of the first anchor point.

*time2*

[in] Time coordinate of the second anchor point.

*ppb*

[in] Pips per bar.

### Returned value

true if successful, false if error.

## PipsPerBar (Get Method)

Gets the value of "Pips per bar" property.

```
double PipsPerBar() const
```

### Returned value

Value of "Pips per bar" property of the object, assigned to the class instance. If there is no object assigned, it returns EMPTY\_VALUE.

## PipsPerBar (Set Method)

Sets new value for "Pips per bar" property.

```
bool PipsPerBar(  
    double ppb // Pips per bar  
)
```

### Parameters

*ppb*

[in] New value for "Pips per bar" property.

### Returned value

true if successful, false if property hasn't changed.



## Save

Saves object parameters to file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] handle of the binary file already opened using by function FileOpen(...).

### Returned value

true if successful, false if error.

## Load

Loads object parameters from file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] handle of the binary file already opened using by function FileOpen(...).

### Returned value

true if successful, false if error.

## Type

Returns graphic object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier (OBJ\_GANNLIN for [CChartObjectGannLine](#)).

## CChartObjectGannFan

Class CChartObjectGannFan is a class for simplified access to "Gann Fan" graphic object properties.

### Description

Class CChartObjectGannFan provides access to "Gann Fan" object properties.

### Declaration

```
class CChartObjectGannFan : public CChartObjectTrend
```

### Title

```
#include <ChartObjects\ChartObjectsGann.mqh>
```

### Class Methods

| Create                       |                                      |
|------------------------------|--------------------------------------|
| <a href="#">Create</a>       | Creates graphic object "Gann Fan"    |
| Properties                   |                                      |
| <a href="#">PipsPerBar</a>   | Gets/Sets property "Pips per bar"    |
| <a href="#">Downtrend</a>    | Gets/Sets property "Downtrend"       |
| Input/output                 |                                      |
| virtual <a href="#">Save</a> | Virtual method for writing to file   |
| virtual <a href="#">Load</a> | Virtual method for reading from file |
| virtual <a href="#">Type</a> | Virtual method of identification     |

### See also

[Object types](#), [Graphic objects](#)

## Create

Creates graphic object "Gann Fan".

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string     name,         // Object name  
    int       window,        // Chart window  
    datetime  time1,         // First time coordinate  
    double    price1,        // First price coordinate  
    datetime  time2,         // Second time coordinate  
    double    ppb            // Pips per bar  
)
```

### Parameters

*chart\_id*

[in] Chart identifier (0 - current chart).

*name*

[in] An unique name of the object to create.

*window*

[in] Chart window number (0 - base window).

*time1*

[in] Time coordinate of the first anchor point.

*price1*

[in] Price coordinate of the first anchor point.

*time2*

[in] Time coordinate of the second anchor point.

*ppb*

[in] Pips per bar.

### Returned value

true if successful, false if error.

## PipsPerBar (Get Method)

Gets the value of "Pips per bar" property.

```
double PipsPerBar() const
```

### Returned value

Value of "Pips per bar" property of the object, assigned to the class instance. If there is no object assigned, it returns EMPTY\_VALUE.

## PipsPerBar (Set Method)

Sets new value for "Pips per bar" property.

```
bool PipsPerBar(  
    double ppb    // Pips per bar  
)
```

### Parameters

*ppb*

[in] New value for "Pips per bar" property.

### Returned value

true if successful, false if property hasn't changed.

## Downtrend (Get Method)

Gets the value of "Downtrend" property.

```
bool Downtrend() const
```

### Returned value

Value of the "Downtrend" property of the object, assigned to the class instance. If there is no object assigned, it returns false.

## Downtrend (Set Method)

Sets new value of "Downtrend" property.

```
bool Downtrend(  
    bool downtrend    // Flag value  
)
```

### Parameters

*downtrend*

[in] New value for "Downtrend" property.

### Returned value

true if successful, false if property hasn't changed.

## Save

Saves object parameters to file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] handle of the binary file already opened using by function FileOpen(...).

### Returned value

true if successful, false if error.



## Load

Loads object parameters from file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] handle of the binary file already opened using by function FileOpen(...).

### Returned value

true if successful, false if error.

## Type

Returns graphic object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier (OBJ\_GANNFAN for [CChartObjectGannFan](#)).

## CChartObjectGannGrid

Class CChartObjectGannGrid is a class for simplified access to "Gann Grid" graphic object properties.

### Description

Class CChartObjectGannGrid provides access to "Gann Grid" object properties.

### Declaration

```
class CChartObjectGannGrid : public CChartObjectTrend
```

### Title

```
#include <ChartObjects\ChartObjectsGann.mqh>
```

### Class Methods

| Create                       |                                      |
|------------------------------|--------------------------------------|
| <a href="#">Create</a>       | Creates graphic object "Gann Grid"   |
| Properties                   |                                      |
| <a href="#">PipsPerBar</a>   | Gets/Sets property "Pips per bar"    |
| <a href="#">Downtrend</a>    | Gets/Sets property "Downtrend"       |
| Input/output                 |                                      |
| virtual <a href="#">Save</a> | Virtual method for writing to file   |
| virtual <a href="#">Load</a> | Virtual method for reading from file |
| virtual <a href="#">Type</a> | Virtual method of identification     |

### See also

[Object types](#), [Graphic objects](#)

## Create

Creates graphic object "Gann Grid".

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string     name,         // Object name  
    int       window,        // Chart window  
    datetime   time1,        // First time coordinate  
    double     price1,       // First price coordinate  
    datetime   time2,        // Second time coordinate  
    double     ppb           // Pips per bar  
)
```

### Parameters

*chart\_id*

[in] Chart identifier (0 - current chart).

*name*

[in] An unique name of the object to create.

*window*

[in] Chart window number (0 - base window).

*time1*

[in] Time coordinate of the first anchor point.

*price1*

[in] Price coordinate of the first anchor point.

*time2*

[in] Time coordinate of the second anchor point.

*ppb*

[in] Pips per bar.

### Returned value

true if successful, false if error.

## PipsPerBar (Get Method)

Gets the value of "Pips per bar" property.

```
double PipsPerBar() const
```

### Returned value

Value of "Pips per bar" property of the object, assigned to the class instance. If there is no object assigned, it returns EMPTY\_VALUE.

## PipsPerBar (Set Method)

Sets new value for "Pips per bar" property.

```
bool PipsPerBar(  
    double ppb    // Pips per bar  
)
```

### Parameters

*ppb*

[in] New value for "Pips per bar" property.

### Returned value

true if successful, false if property hasn't changed.

## Downtrend (Get Method)

Gets the value of "Downtrend" property.

```
bool Downtrend() const
```

### Returned value

Value of "Downtrend" property of the object, assigned to the class instance. If there is no object assigned, it returns false.

## Downtrend (Set Method)

Sets new value of "Downtrend" property.

```
bool Downtrend(  
    bool downtrend    // Flag value  
)
```

### Parameters

*downtrend*

[in] New value for "Downtrend" property.

### Returned value

true if successful, false if property hasn't changed.

## Save

Saves object parameters to file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] handle of the binary file already opened using by function FileOpen(...).

### Returned value

true if successful, false if error.

## Load

Loads object parameters from file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] handle of the binary file already opened using by function FileOpen(...).

### Returned value

true if successful, false if error.



## Type

Returns graphic object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier (OBJ\_GANNGRID for [CChartObjectGannGrid](#)).

## Fibonacci Tools

A group of graphic objects "Fibonacci Tools".

This section contains the technical details of working with a group of classes of graphical objects "Fibonacci Tools" and a description of the relevant components of the MQL5 Standard Library .

| Class name                                       | Object                                 |
|--------------------------------------------------|----------------------------------------|
| <a href="#"><u>CChartObjectFibo</u></a>          | Graphic object "Fibonacci Retracement" |
| <a href="#"><u>CChartObjectFiboTimes</u></a>     | Graphic object "Fibonacci Time Zones"  |
| <a href="#"><u>CChartObjectFiboFan</u></a>       | Graphic object "Fibonacci Fan"         |
| <a href="#"><u>CChartObjectFiboArc</u></a>       | Graphic object "Fibonacci Arc"         |
| <a href="#"><u>CChartObjectFiboChannel</u></a>   | Graphic object "Fibonacci Channel"     |
| <a href="#"><u>CChartObjectFiboExpansion</u></a> | Graphic object "Fibonacci Expansion"   |

### See also

[Object types](#), [Graphic objects](#)

# CChartObjectFibo

Class CChartObjectFibo is a class for simplified access to "Fibonacci Retracement" graphic object properties.

## Description

Class CChartObjectFibo provides access to "Fibonacci Retracement" object properties.

## Declaration

```
class CChartObjectFibo : public CChartObjectTrend
```

## Title

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

## Class Methods

| Create                       |                                                |
|------------------------------|------------------------------------------------|
| <a href="#">Create</a>       | Creates graphic object "Fibonacci Retracement" |
| Input/output                 |                                                |
| virtual <a href="#">Type</a> | Virtual method of identification               |

### See also

[Object types](#), [Graphic objects](#)

## Create

Creates graphic object "Fibonacci Retracement".

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string    name,         // Object name  
    int       window,       // Chart window  
    datetime  time1,        // First time coordinate  
    double    price1,       // First price coordinate  
    datetime  time2,        // Second time coordinate  
    double    price2        // Second price coordinate  
)
```

### Parameters

*chart\_id*

[in] Chart identifier (0 - current chart).

*name*

[in] An unique name of the object to create.

*window*

[in] Chart window number (0 - base window).

*time1*

[in] Time coordinate of the first anchor point.

*price1*

[in] Price coordinate of the first anchor point.

*time2*

[in] Time coordinate of the second anchor point.

*price2*

[in] Price coordinate of the second anchor point.

### Returned value

true if successful, false if error.

## Type

Returns graphic object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier (OBJ\_FIBO for [CChartObjectFibo](#)).

# CChartObjectFiboTimes

Class CChartObjectFiboTimes is a class for simplified access to "Fibonacci Time Zones" graphic object properties.

## Description

Class CChartObjectFiboTimes provides access to "Fibonacci Time Zones" object properties.

## Declaration

```
class CChartObjectFiboTimes : public CChartObject
```

## Title

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

## Class Methods

| Create              |                                               |
|---------------------|-----------------------------------------------|
| <u>Create</u>       | Creates graphic object "Fibonacci Time Zones" |
| Input/output        |                                               |
| virtual <u>Type</u> | Virtual method of identification              |

### See also

[Object types](#), [Graphic objects](#)

## Create

Creates graphic object "Fibonacci Time Zones".

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string     name,         // Object name  
    int       window,        // Chart window  
    datetime   time1,        // First time coordinate  
    double     price1,       // First price coordinate  
    datetime   time2,        // Second time coordinate  
    double     price2        // Second price coordinate  
)
```

### Parameters

*chart\_id*

[in] Chart identifier (0 - current chart).

*name*

[in] An unique name of the object to create.

*window*

[in] Chart window number (0 - base window).

*time1*

[in] Time coordinate of the first anchor point.

*price1*

[in] Price coordinate of the first anchor point.

*time2*

[in] Time coordinate of the second anchor point.

*price2*

[in] Price coordinate of the second anchor point.

### Returned value

true if successful, false if error.

## Type

Returns graphic object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier (OBJ\_FIBOTIMES for [CChartObjectFiboTimes](#)).



# CChartObjectFiboFan

Class CChartObjectFiboFan is a class for simplified access to "Fibonacci Fan" graphic object properties.

## Description

Class CChartObjectFiboFan provides access to "Fibonacci Fan" object properties.

## Declaration

```
class CChartObjectFiboFan : public CChartObject
```

## Title

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

## Class Methods

| Create              |                                        |
|---------------------|----------------------------------------|
| <u>Create</u>       | Creates graphic object "Fibonacci Fan" |
| Input/output        |                                        |
| virtual <u>Type</u> | Virtual method of identification       |

### See also

[Object types](#), [Graphic objects](#)

## Create

Creates graphic object "Fibonacci Fan".

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string     name,         // Object name  
    int       window,        // Chart window  
    datetime  time1,         // First time coordinate  
    double    price1,        // First price coordinate  
    datetime  time2,         // Second time coordinate  
    double    price2         // Second price coordinate  
)
```

### Parameters

*chart\_id*

[in] Chart identifier (0 - current chart).

*name*

[in] An unique name of the object to create.

*window*

[in] Chart window number (0 - base window).

*time1*

[in] Time coordinate of the first anchor point.

*price1*

[in] Price coordinate of the first anchor point.

*time2*

[in] Time coordinate of the second anchor point.

*price2*

[in] Price coordinate of the second anchor point.

### Returned value

true if successful, false if error.

## Type

Returns graphic object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier (OBJ\_FIBOFAN for [CChartObjectFiboFan](#)).

# CChartObjectFiboArc

Class CChartObjectFiboArc is a class for simplified access to "Fibonacci Arc" graphic object properties.

## Description

Class CChartObjectFiboArc provides access to "Fibonacci Arc" object properties.

## Declaration

```
class CChartObjectFiboArc : public CChartObject
```

## Title

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

## Class Methods

| Create                       |                                        |
|------------------------------|----------------------------------------|
| <a href="#">Create</a>       | Creates graphic object "Fibonacci Arc" |
| Properties                   |                                        |
| <a href="#">Scale</a>        | Gets/Sets property "Scale"             |
| <a href="#">Ellipse</a>      | Gets/Sets property "Ellipse"           |
| Input/output                 |                                        |
| virtual <a href="#">Save</a> | Virtual method for writing to file     |
| virtual <a href="#">Load</a> | Virtual method for reading from file   |
| virtual <a href="#">Type</a> | Virtual method of identification       |

### See also

[Object types](#), [Graphic objects](#)

## Create

Creates graphic object "Fibonacci Arc"

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string    name,          // Object name  
    int       window,        // Chart window  
    datetime  time1,         // First time coordinate  
    double    price1,        // First price coordinate  
    datetime  time2,         // Second time coordinate  
    double    price2,        // Second price coordinate  
    double    scale          // Scale  
)
```

### Parameters

*chart\_id*

[in] Chart identifier (0 - current chart).

*name*

[in] An unique name of the object to create.

*window*

[in] Chart window number (0 - base window).

*time1*

[in] Time coordinate of the first anchor point.

*price1*

[in] Price coordinate of the first anchor point.

*time2*

[in] Time coordinate of the second anchor point.

*price2*

[in] Price coordinate of the second anchor point.

*scale*

[in] Scale.

### Returned value

true if successful, false if error.

## Scale (Get Method)

Gets the value of "Scale" property.

```
double Scale() const
```

### Returned value

Value of "Scale" property of the object, assigned to the class instance. If there is no object assigned, it returns EMPTY\_VALUE.

## Scale (Set Method)

Sets new value for "Scale" property.

```
bool Scale(  
    double scale    // Scale  
)
```

### Parameters

*scale*

[in] New value for "Scale" property.

### Returned value

true if successful, false if property hasn't changed.

## Ellipse (Get Method)

Gets the value of "Ellipse" property.

```
bool Ellipse() const
```

### Returned value

Value of "Ellipse" property of the object, assigned to the class instance. If there is no object assigned, it returns false.

## Ellipse (Set Method)

Sets new flag value for "Ellipse" property.

```
bool Ellipse(  
    bool ellipse    // flag value  
)
```

### Parameters

*ellipse*

[in] New value for "Scale" property.

### Returned value

true if successful, false if property hasn't changed.

## Save

Saves object parameters to file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] handle of the binary file already opened using by function FileOpen(...).

### Returned value

true if successful, false if error.



## Load

Loads object parameters from file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] handle of the binary file already opened using by function FileOpen(...).

### Returned value

true if successful, false if error.

## Type

Returns graphic object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier (OBJ\_FIBOARC for [CChartObjectFiboArc](#)).

# CChartObjectFiboChannel

Class CChartObjectFiboChannel is a class for simplified access to "Fibonacci Channel" graphic object properties.

## Description

Class CChartObjectFiboChannel provides access to "Fibonacci Channel" object properties.

## Declaration

```
class CChartObjectFiboChannel : public CChartObjectTrend
```

## Title

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

## Class Methods

| Create                       |                                            |
|------------------------------|--------------------------------------------|
| <a href="#">Create</a>       | Creates graphic object "Fibonacci Channel" |
| Input/output                 |                                            |
| virtual <a href="#">Type</a> | Virtual method of identification           |

### See also

[Object types](#), [Graphic objects](#)

## Create

Creates graphic object "Fibonacci Channel".

```
bool Create(  
    long      chart_id,    // Chart identifier  
    string     name,       // Object name  
    int       window,      // Chart window  
    datetime  time1,       // First time coordinate  
    double    price1,      // First price coordinate  
    datetime  time2,       // Second time coordinate  
    double    price2,      // Second price coordinate  
    datetime  time3,       // Third time coordinate  
    double    price3       // Third price coordinate  
)
```

### Parameters

*chart\_id*

[in] Chart identifier (0 - current chart).

*name*

[in] An unique name of the object to create.

*window*

[in] Chart window number (0 - base window).

*time1*

[in] Time coordinate of the first anchor point.

*price1*

[in] Price coordinate of the first anchor point.

*time2*

[in] Time coordinate of the second anchor point.

*price2*

[in] Price coordinate of the second anchor point.

*time3*

[in] Time coordinate of the third anchor point.

*price3*

[in] Price coordinate of the third anchor point.

### Returned value

true if successful, false if error.

## Type

Returns graphic object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier (OBJ\_FIBOCHANNEL for [CChartObjectFiboChannel](#)).

# CChartObjectFiboExpansion

Class CChartObjectFiboExpansion is a class for simplified access to "Fibonacci Expansion" graphic object properties.

## Description

Class CChartObjectFiboExpansion provides access to "Fibonacci Expansion" object properties.

## Declaration

```
class CChartObjectFiboExpansion : public CChartObjectTrend
```

## Title

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

## Class Methods

| Create              |                                              |
|---------------------|----------------------------------------------|
| <u>Create</u>       | Creates graphic object "Fibonacci Expansion" |
| Input/output        |                                              |
| virtual <u>Type</u> | Virtual method of identification             |

### See also

[Object types](#), [Graphic objects](#)

## Create

Creates graphic object "Fibonacci Expansion".

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string     name,         // Object name  
    int       window,        // Chart window  
    datetime   time1,        // First time coordinate  
    double     price1,       // First price coordinate  
    datetime   time2,        // Second time coordinate  
    double     price2,       // Second price coordinate  
    datetime   time3,        // Third time coordinate  
    double     price3        // Third price coordinate  
)
```

### Parameters

*chart\_id*

[in] Chart identifier (0 - current chart).

*name*

[in] An unique name of the object to create.

*window*

[in] Chart window number (0 - base window).

*time1*

[in] Time coordinate of the first anchor point.

*price1*

[in] Price coordinate of the first anchor point.

*time2*

[in] Time coordinate of the second anchor point.

*price2*

[in] Price coordinate of the second anchor point.

*time3*

[in] Time coordinate of the third anchor point.

*price3*

[in] Price coordinate of the third anchor point.

### Returned value

true if successful, false if error.

## Type

Returns graphic object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier (OBJ\_EXPANSION for [CChartObjectFiboExpansion](#)).



## Elliott Tools

A group of graphic objects "Elliott Tools".

This section contains the technical details of working with a group of classes of graphical objects "Elliott Tools" and a description of the relevant components of the MQL5 Standard Library .

| Class name                                      | Object                           |
|-------------------------------------------------|----------------------------------|
| <a href="#"><u>CChartObjectElliottWave3</u></a> | Graphic object "Correcting Wave" |
| <a href="#"><u>CChartObjectElliottWave5</u></a> | Graphic object "Impulse Wave"    |

See also

[Object types](#), [Graphic objects](#)

## CChartObjectElliottWave3

Class CChartObjectElliottWave3 is a class for simplified access to "Correcting Wave" graphic object properties.

### Description

Class CChartObjectElliottWave3 provides access to "Correcting Wave" object properties.

### Declaration

```
class CChartObjectElliottWave3 : public CChartObject
```

### Title

```
#include <ChartObjects\ChartObjectsElliott.mqh>
```

### Class Methods

| Create                       |                                          |
|------------------------------|------------------------------------------|
| <a href="#">Create</a>       | Creates graphic object "Correcting Wave" |
| Properties                   |                                          |
| <a href="#">Degree</a>       | Gets/Sets property "Degree"              |
| <a href="#">Lines</a>        | Gets/Sets property "Lines"               |
| Input/output                 |                                          |
| virtual <a href="#">Save</a> | Virtual method for writing to file       |
| virtual <a href="#">Load</a> | Virtual method for reading from file     |
| virtual <a href="#">Type</a> | Virtual method of identification         |

Derived classes:

- [CChartObjectElliottWave5](#)

See also

[Object types](#), [Graphic objects](#)

## Create

Creates graphic object "Correcting Wave".

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string     name,         // Object name  
    int        window,       // Chart window  
    datetime   time1,        // First time coordinate  
    double     price1,       // First price coordinate  
    datetime   time2,        // Second time coordinate  
    double     price2,       // Second price coordinate  
    datetime   time3,        // Third time coordinate  
    double     price3        // Third price coordinate  
)
```

### Parameters

*chart\_id*

[in] Chart identifier (0 - current chart).

*name*

[in] An unique name of the object to create.

*window*

[in] Chart window number (0 - base window).

*time1*

[in] Time coordinate of the first anchor point.

*price1*

[in] Price coordinate of the first anchor point.

*time2*

[in] Time coordinate of the second anchor point.

*price2*

[in] Price coordinate of the second anchor point.

*time3*

[in] Time coordinate of the third anchor point.

*price3*

[in] Time coordinate of the third anchor point.

### Returned value

true if successful, false if error.

## Degree (Get Method)

Gets the value of "Degree" property.

```
ENUM_ELLIOT_WAVE_DEGREE Degree() const
```

### Returned value

Value of "Degree" property of the object, assigned to the class instance. If there is no object assigned, it returns WRONG\_VALUE.

## Degree (Set Method)

Sets new value for "Degree" property.

```
bool Degree(  
    ENUM_ELLIOT_WAVE_DEGREE degree // property value  
)
```

### Parameters

*degree*

[in] New value for "Degree" property.

### Returned value

true if successful, false if property hasn't changed.

## Lines (Get Method)

Gets the value of "Lines" property.

```
bool Lines() const
```

### Returned value

Value of "Lines" property of the object, assigned to the class instance. If there is no object assigned, it returns false.

## Lines (Set Method)

Sets new value for "Lines" property.

```
bool Lines(  
    bool lines    // flag value  
)
```

### Parameters

*lines*

[in] New value for "Lines" property.

### Returned value

true if successful, false if flag hasn't changed.

## Save

Saves object parameters to file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] handle of the binary file already opened using by function FileOpen(...).

### Returned value

true if successful, false if error.

## Load

Loads object parameters from file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] handle of the binary file already opened using by function FileOpen(...).

### Returned value

true if successful, false if error.

## Type

Returns graphic object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier (OBJ\_ELLIOTWAVE3 for [CChartObjectElliottWave3](#)).



## CChartObjectElliottWave5

Class CChartObjectElliottWave5 is a class for simplified access to "Impulse Wave" graphic object properties.

### Description

Class CChartObjectElliottWave5 provides access to "Impulse Wave" object properties.

### Declaration

```
class CChartObjectElliottWave5 : public CChartObjectElliottWave3
```

### Title

```
#include <ChartObjects\ChartObjectsElliott.mqh>
```

### Class Methods

| Create              |                                       |
|---------------------|---------------------------------------|
| <u>Create</u>       | Creates graphic object "Impulse Wave" |
| Input/output        |                                       |
| virtual <u>Type</u> | Virtual method of identification      |

### See also

[Object types](#), [Graphic objects](#)

## Create

Creates graphic object "Impulse Wave".

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string     name,         // Object name  
    int        window,       // Chart window  
    datetime   time1,        // First time coordinate  
    double     price1,        // First price coordinate  
    datetime   time2,        // Second time coordinate  
    double     price2,        // Second price coordinate  
    datetime   time3,        // Third time coordinate  
    double     price3,        // Third price coordinate  
    datetime   time4,        // Fourth time coordinate  
    double     price4,        // Fourth price coordinate  
    datetime   time5,        // Fifth time coordinate  
    double     price5,        // Fifth price coordinate  
)
```

### Parameters

*chart\_id*

[in] Chart identifier (0 - current chart).

*name*

[in] An unique name of the object to create.

*window*

[in] Chart window number (0 - base window).

*time1*

[in] Time coordinate of the first anchor point.

*price1*

[in] Price coordinate of the first anchor point.

*time2*

[in] Time coordinate of the second anchor point.

*price2*

[in] Price coordinate of the second anchor point.

*time3*

[in] Time coordinate of the third anchor point.

*price3*

[in] Price coordinate of the third anchor point.

*time4*

[in] Time coordinate of the fourth anchor point.

*price4*

[in] Price coordinate of the fourth anchor point.

*time5*

[in] Time coordinate of the fifth anchor point.

*price5*

[in] Price coordinate of the fifth anchor point.

#### Returned value

true if successful, false if error.

## Type

Returns graphic object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier (OBJ\_ELLIOTWAVE5 for [CChartObjectElliottWave5](#)).

## Objects Shapes

A group of graphic objects "Shapes".

This section contains the technical details of working with a group of classes of graphical objects "Shapes" and a description of the relevant components of the MQL5 Standard Library .

| Class name                                   | Object                     |
|----------------------------------------------|----------------------------|
| <a href="#"><u>CChartObjectRectangle</u></a> | Graphic object "Rectangle" |
| <a href="#"><u>CChartObjectTriangle</u></a>  | Graphic object "Triangle"  |
| <a href="#"><u>CChartObjectEllipse</u></a>   | Graphic object "Ellipse"   |

See also

[Object types](#), [Graphic objects](#)

# CChartObjectRectangle

Class CChartObjectRectangle is a class for simplified access to "Rectangle" graphic object properties.

## Description

Class CChartObjectRectangle provides access to "Rectangle" object properties.

## Declaration

```
class CChartObjectRectangle : public CChartObject
```

## Title

```
#include <ChartObjects\ChartObjectsShapes.mqh>
```

## Class Methods

|                              |                                    |
|------------------------------|------------------------------------|
| Create                       |                                    |
| <a href="#">Create</a>       | Creates graphic object "Rectangle" |
| Input/output                 |                                    |
| virtual <a href="#">Type</a> | Virtual method of identification   |

See also

[Object types](#), [Graphic objects](#)

## Create

Creates graphic object "Rectangle".

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string    name,          // Object name  
    long      window,        // Chart window  
    datetime  time1,         // First time coordinate  
    double    price1,        // First price coordinate  
    datetime  time2,         // Second time coordinate  
    double    price2         // Second price coordinate  
)
```

### Parameters

*chart\_id*

[in] Chart identifier (0 - current chart).

*name*

[in] An unique name of the object to create.

*window*

[in] Chart window number (0 - base window).

*time1*

[in] Time coordinate of the first anchor point.

*price1*

[in] Price coordinate of the first anchor point.

*time2*

[in] Time coordinate of the second anchor point.

*price2*

[in] Price coordinate of the second anchor point.

### Returned value

true if successful, 0 if error

## Type

Returns object type identifier of the graphic object.

```
int Type() const
```

### Returned value

Object type identifier (OBJ\_RECTANGLE for [CChartObjectRectangle](#)).



# CChartObjectTriangle

Class CChartObjectTriangle is a class for simplified access to "Triangle" graphic object properties.

## Description

Class CChartObjectTriangle provides access to "Triangle" object properties.

## Declaration

```
class CChartObjectTriangle : public CChartObject
```

## Title

```
#include <ChartObjects\ChartObjectsShapes.mqh>
```

## Class Methods

|                              |                                   |
|------------------------------|-----------------------------------|
| Create                       |                                   |
| <a href="#">Create</a>       | Creates graphic object "Triangle" |
| Input/output                 |                                   |
| virtual <a href="#">Type</a> | Virtual method of identification  |

See also

[Object types](#), [Graphic objects](#)

## Create

Creates graphic object "Triangle".

```
bool Create(  
    long      chart_id,      // Chart identifier  
    string    name,          // Object name  
    long      window,        // Chart window  
    datetime  time1,         // First time coordinate  
    double    price1,        // First price coordinate  
    datetime  time2,         // Second time coordinate  
    double    price2,        // Second price coordinate  
    datetime  time3,         // Third time coordinate  
    double    price3         // Third price coordinate  
)
```

### Parameters

*chart\_id*

[in] Chart identifier (0 - current chart).

*name*

[in] An unique name of the object to create.

*window*

[in] Chart window number (0 - base window).

*time1*

[in] Time coordinate of the first anchor point.

*price1*

[in] Price coordinate of the first anchor point.

*time2*

[in] Time coordinate of the second anchor point.

*price2*

[in] Price coordinate of the second anchor point.

*time3*

[in] Time coordinate of the third anchor point.

*price3*

[in] Price coordinate of the third anchor point.

### Returned value

true if successful, false if error

## Type

Returns object type identifier of the graphic object.

```
int Type() const
```

### Returned value

Object type identifier (OBJ\_TRIANGLE for [CChartObjectTriangle](#)).

# CChartObjectEllipse

Class CChartObjectEllipse is a class for simplified access to "Ellipse" graphic object properties.

## Description

Class CChartObjectEllipse provides access to "Ellipse" object properties.

## Declaration

```
class CChartObjectEllipse : public CChartObject
```

## Title

```
#include <ChartObjects\ChartObjectsShapes.mqh>
```

## Class Methods

|                              |                                  |
|------------------------------|----------------------------------|
| Create                       |                                  |
| <a href="#">Create</a>       | Creates graphic object "Ellipse" |
| Input/output                 |                                  |
| virtual <a href="#">Type</a> | Virtual method of identification |

### See also

[Object types](#), [Graphic objects](#)

## Create

Creates graphic object "Ellipse".

```
bool Create(  
    long      chart_id,    // Chart identifier  
    string    name,        // Object name  
    int       window,      // Chart window  
    datetime  time1,       // First time coordinate  
    double    price1,      // First price coordinate  
    datetime  time2,       // Second time coordinate  
    double    price2,      // Second price coordinate  
    datetime  time3,       // Third time coordinate  
    double    price3       // Third price coordinate  
)
```

### Parameters

*chart\_id*

[in] Chart identifier (0 - current chart).

*name*

[in] An unique name of the object to create.

*window*

[in] Chart window number (0 - base window).

*time1*

[in] Time coordinate of the first anchor point.

*price1*

[in] Price coordinate of the first anchor point.

*time2*

[in] Time coordinate of the second anchor point.

*price2*

[in] Price coordinate of the second anchor point.

*time3*

[in] Time coordinate of the third anchor point.

*price3*

[in] Price coordinate of the third anchor point.

### Returned value

true if successful, false if error

## Type

Returns object type identifier of the graphic object.

```
int Type() const
```

### Returned value

Object type identifier (OBJ\_ELLIPSE for [CChartObjectEllipse](#)).

## Objects Arrows

Group for graphic objects Arrows.

This section contains the technical details of working with a group of classes of graphical objects "Arrow" and a description of the relevant components of the MQL5 Standard Library . In essence, the arrow - this is some icon to be displayed to the user, which matches a certain code. There are two types of graphical objects "Arrow" to display icons on the charts:

- Object "Arrow", which allows you to specify the code icon displayed object.
- Group objects to display certain types of icons (and the corresponding certain fixed code).

### Class for the arrow displays icons of arbitrary code

| Class name                               | Name of the object arrow |
|------------------------------------------|--------------------------|
| <a href="#"><u>CChartObjectArrow</u></a> | Arrow                    |

### Classes for the arrow icon fixed code

| Class name                                         | Name of the object arrow |
|----------------------------------------------------|--------------------------|
| <a href="#"><u>CChartObjectArrowCheck</u></a>      | Check                    |
| <a href="#"><u>CChartObjectArrowDown</u></a>       | Arrow Up                 |
| <a href="#"><u>CChartObjectArrowUp</u></a>         | Arrow Down               |
| <a href="#"><u>CChartObjectArrowStop</u></a>       | Stop Sign                |
| <a href="#"><u>CChartObjectArrowThumbDown</u></a>  | Thumbs Up                |
| <a href="#"><u>CChartObjectArrowThumbUp</u></a>    | Thumbs Down              |
| <a href="#"><u>CChartObjectArrowLeftPrice</u></a>  | Left Price Label         |
| <a href="#"><u>CChartObjectArrowRightPrice</u></a> | Right Price Label        |

See also

[Object types](#), [Methods of binding sites](#), [Graphic objects](#)

# CChartObjectArrow

Class CChartObjectArrow is a class for simplified access to "Arrow" graphic object properties.

## Description

Class CChartObjectArrow provides access to common properties of objects "Arrow" to all of its descendants.

## Declaration

```
class CChartObjectArrow : public CChartObject
```

## Title

```
#include <ChartObjects\ChartObjectsArrows.mqh>
```

## Class Methods

| Create              |                                      |
|---------------------|--------------------------------------|
| <u>Create</u>       | Creates graphic object "Arrow"       |
| Properties          |                                      |
| <u>ArrowCode</u>    | Gets/Sets property "Arrow Code"      |
| <u>Anchor</u>       | Gets/Sets property "Anchor"          |
| Input/output        |                                      |
| virtual <u>Save</u> | Virtual method for writing to file   |
| virtual <u>Load</u> | Virtual method for reading from file |
| virtual <u>Type</u> | Virtual method of identification     |

## See also

[Object types](#), [Methods of binding sites](#), [Graphic objects](#)



## Create

Creates graphic object "Arrow".

```
bool Create(  
    long      chart_id,      // Chart ID  
    string     name,         // Object Name  
    int       window,        // Chart Window  
    datetime  time,          // Time  
    double    price,         // Price  
    char      code           // Arrow Code  
)
```

### Parameters

*chart\_id*

[in] Chart Identifier (0 - current chart).

*name*

[in] Object name (Should be unique).

*window*

[in] Chart window number (0 - base window).

*time*

[in] Time coordinate.

*price*

[in] Price coordinate.

*code*

[in] "Arrow" code (Wingdings).

### Returned value

true - if successful, otherwise false.

### Example:

```
///--- example for CChartObjectArrow::Create  
#include <ChartObjects\ChartObjectsArrows.mqh>  
///---  
void OnStart()  
{  
    CChartObjectArrow arrow;  
    ///--- set object parameters  
    double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);  
    if(!arrow.Create(0,"Arrow",0,TimeCurrent(),price,181))  
    {  
        ///--- arrow create error  
        printf("Arrow create: Error %d!",GetLastError());  
        ///---  
    }  
}
```

```
        return;  
    }  
    //--- use arrow  
    //--- . . .  
}
```

## ArrowCode (Get Method)

Gets code of the symbol for "Arrow".

```
char ArrowCode() const
```

### Returned value

Symbol code of "Arrow" object, assigned to the class instance. If there is no object assigned, it returns 0.

## ArrowCode (Set Method)

Sets symbol code for "Arrow"

```
bool ArrowCode (
    char code      // Code value
)
```

### Parameters

*code*

[in] new value for "arrow" code (Wingdings).

### Returned value

true - if successful, false - if code hasn't changed.

### Example:

```
//--- example for CChartObjectArrow::ArrowCode
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
{
    CChartObjectArrow arrow;
    char code=181;
    //--- set object parameters
    double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
    if(!arrow.Create(0,"Arrow",0,TimeCurrent(),price,code))
    {
        //--- arrow create error
        printf("Arrow create: Error %d!",GetLastError());
        //---
        return;
    }
    //--- use arrow to possible changes code
    //--- . . .
    //--- get code of arrow
    if(arrow.ArrowCode()!=code)
    {
        //--- set code of arrow
```

```
        arrow.ArrowCode(code);  
    }  
    //--- use arrow  
    //--- . . .  
}
```

## Anchor (Get Method)

Gets anchor type of the "Arrow" object

```
ENUM_ARROW_ANCHOR Anchor() const
```

### Returned value

Anchor type of "Arrow" object, assigned to the class instance. If there is no object assigned, it returns WRONG\_VALUE.

## Anchor (Set Method)

Sets color for graphic object

```
bool Anchor(
    ENUM_ARROW_ANCHOR new_color    // new color
)
```

### Parameters

*new\_color*

[in] New value of color for line of the graphic object.

### Returned value

true if successful, false if color hasn't changed.

### Example:

```
//--- example for CChartObject::Anchor
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
{
    CChartObjectArrow arrow;
    ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM;
    //--- set object parameters
    double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
    if(!arrow.Create(0,"Arrow",0,TimeCurrent(),price,181))
    {
        //--- arrow create error
        printf("Arrow create: Error %d!",GetLastError());
        //---
        return;
    }
    //--- get anchor of arrow
    if(arrow.Anchor()!=anchor)
    {
        //--- set anchor of arrow
        arrow.Anchor(anchor);
    }
    //--- use arrow
```

```
//--- . . .  
}
```

## Save

Saves object parameters to file.

```
virtual bool Save(  
    int file_handle    // file handle  
)
```

### Parameters

*file\_handle*

[in] handle of the file already opened using by function FileOpen(...).

### Returned value

true if successful, otherwise false.

### Example:

```
//--- example for CChartObjectArrow::Save  
#include <ChartObjects\ChartObjectsArrows.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CChartObjectArrow arrow;  
    //--- set object parameters  
    double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);  
    if(!arrow.Create(0,"Arrow",0,TimeCurrent(),price,181))  
    {  
        //--- arrow create error  
        printf("Arrow create: Error %d!",GetLastError());  
        //---  
        return;  
    }  
    //--- open file  
    file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!arrow.Save(file_handle))  
        {  
            //--- file save error  
            printf("File save: Error %d!",GetLastError());  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }  
}
```

## Load

Loads object parameters from file.

```
virtual bool Load(  
    int file_handle    // file handle  
)
```

### Parameters

*file\_handle*

[in] handle of the file already opened using by function FileOpen(...).

### Returned value

true if successful, otherwise false.

### Example:

```
//--- example for CChartObjectArrow::Load  
#include <ChartObjects\ChartObjectsArrows.mqh>  
//---  
void OnStart()  
{  
    int file_handle;  
    CChartObjectArrow arrow;  
    //--- open file  
    file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);  
    if(file_handle>=0)  
    {  
        if(!arrow.Load(file_handle))  
        {  
            //--- file load error  
            printf("File load: Error %d!",GetLastError());  
            FileClose(file_handle);  
            //---  
            return;  
        }  
        FileClose(file_handle);  
    }  
    //--- use arrow  
    //--- . . .  
}
```



## Type

Returns graphic object type identifier

```
virtual int Type() const
```

### Returned value

Object type identifier (for example, OBJ\_ARROW for [CChartObjectArrow](#))

### Example:

```
//--- example for CChartObjectArrow::Type
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
{
    CChartObjectArrow arrow;
    //--- get arrow type
    int type=arrow.Type();
}
```

## Arrows with fixed code

Classes "Arrows with fixed code" are classes for simplified access to the properties of the following graphic objects:

| Class name                  | Arrow object name         |
|-----------------------------|---------------------------|
| CChartObjectArrowCheck      | "Arrow Check"             |
| CChartObjectArrowDown       | "Arrow Down"              |
| CChartObjectArrowUp         | "Arrow Up"                |
| CChartObjectArrowStop       | "Arrow Stop"              |
| CChartObjectArrowThumbDown  | "Good" ("Big finger up")  |
| CChartObjectArrowThumbUp    | "Bad" ("Big finger down") |
| CChartObjectArrowLeftPrice  | "Left price" arrow        |
| CChartObjectArrowRightPrice | "Right price" arrow       |

### Description

Classes "Arrows with fixed code" provides access to the object properties.

### Declarations

```

class CChartObjectArrowCheck      : public CChartObjectArrow;
class CChartObjectArrowDown      : public CChartObjectArrow;
class CChartObjectArrowUp        : public CChartObjectArrow;
class CChartObjectArrowStop      : public CChartObjectArrow;
class CChartObjectArrowThumbDown : public CChartObjectArrow;
class CChartObjectArrowThumbUp   : public CChartObjectArrow;
class CChartObjectArrowLeftPrice : public CChartObjectArrow;
class CChartObjectArrowRightPrice : public CChartObjectArrow;

```

### Title

<ChartObjects\ChartObjectsArrows.mqh>

### Class Methods

| Create                       |                                  |
|------------------------------|----------------------------------|
| <a href="#">Create</a>       | Creates graphic object specified |
| Properties                   |                                  |
| <a href="#">ArrowCode</a>    | "Gag" for method of code change  |
| Input/output                 |                                  |
| virtual <a href="#">Type</a> | Virtual method of identification |

See also

[Object types](#), [Methods of binding sites](#), [Graphic objects](#)

## Create

Creates graphic object "Arrow with fixed code".

```
bool Create(
    long      chart_id,      // Chart ID
    string     name,         // Object Name
    int       window,        // Chart Window
    datetime   time,         // Time
    double     price         // Price
)
```

### Parameters

*chart\_id*

[in] Chart identifier (0 - current chart).

*name*

[in] Unique name of the object to create.

*window*

[in] Chart window number (0 - base window).

*time*

[in] Time coordinate.

*price*

[in] Price coordinate.

### Returned value

true if successful, false if error.

### Example:

```
//--- example for CChartObjectArrowCheck::Create
//--- example for CChartObjectArrowDown::Create
//--- example for CChartObjectArrowUp::Create
//--- example for CChartObjectArrowStop::Create
//--- example for CChartObjectArrowThumbDown::Create
//--- example for CChartObjectArrowThumbUp::Create
//--- example for CChartObjectArrowLeftPrice::Create
//--- example for CChartObjectArrowRightPrice::Create
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
{
    //--- for example, take CChartObjectArrowCheck
    CChartObjectArrowCheck arrow;
    //--- set object parameters
    double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
    if(!arrow.Create(0,"ArrowCheck",0,TimeCurrent(),price))
```

```
{
    //--- arrow create error
    printf("Arrow create: Error %d!", GetLastError());
    //---
    return;
}
//--- use arrow
//--- . . .
}
```

## ArrowCode

Prohibits code changes for "Arrow".

```
bool ArrowCode(  
    char code    // code value  
)
```

### Parameters

*code*

[in] any value

### Returned value

Always false.

### Example:

```
//--- example for CChartObjectArrowCheck::ArrowCode  
//--- example for CChartObjectArrowDown::ArrowCode  
//--- example for CChartObjectArrowUp::ArrowCode  
//--- example for CChartObjectArrowStop::ArrowCode  
//--- example for CChartObjectArrowThumbDown::ArrowCode  
//--- example for CChartObjectArrowThumbUp::ArrowCode  
//--- example for CChartObjectArrowLeftPrice::ArrowCode  
//--- example for CChartObjectArrowRightPrice::ArrowCode  
#include <ChartObjects\ChartObjectsArrows.mqh>  
//---  
void OnStart()  
{  
    //--- for example, take CChartObjectArrowCheck  
    CChartObjectArrowCheck arrow;  
    //--- set object parameters  
    double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);  
    if(!arrow.Create(0,"ArrowCheck",0,TimeCurrent(),price))  
    {  
        //--- arrow create error  
        printf("Arrow create: Error %d!",GetLastError());  
        //---  
        return;  
    }  
    //--- set code of arrow  
    if(!arrow.ArrowCode(181))  
    {  
        //--- it is not error  
        printf("Arrow code can not be changed");  
    }  
    //--- use arrow  
    //--- . . .  
}
```

## Type

Returns graphic object type identifier

```
virtual int Type() const
```

### Returned value

Object type identifier (OBJ\_ARROW\_CHECK for CChartObjectArrowCheck, OBJ\_ARROW\_DOWN for CChartObjectArrowDown, OBJ\_ARROW\_UP for CChartObjectArrowUp, OBJ\_ARROW\_STOP for CChartObjectArrowStop, OBJ\_ARROW\_THUMB\_DOWN for CChartObjectArrowThumbDown, OBJ\_ARROW\_THUMB\_UP for CChartObjectArrowThumbUp, OBJ\_ARROW\_LEFT\_PRICE for CChartObjectArrowLeftPrice, OBJ\_ARROW\_RIGHT\_PRICE for CChartObjectArrowRightPrice).

### Example:

```
//--- example for CChartObjectArrowCheck::Type
//--- example for CChartObjectArrowDown::Type
//--- example for CChartObjectArrowUp::Type
//--- example for CChartObjectArrowStop::Type
//--- example for CChartObjectArrowThumbDown::Type
//--- example for CChartObjectArrowThumbUp::Type
//--- example for CChartObjectArrowLeftPrice::Type
//--- example for CChartObjectArrowRightPrice::Type
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
{
//--- for example, take CChartObjectArrowCheck
    CChartObjectArrowCheck arrow;
//--- get arrow type
    int type=arrow.Type();
}
```

## Object Controls

A group of graphic objects "Object Controls".

This section contains the technical details of working with a group of classes of graphical objects "Object Controls" and a description of the relevant components of the MQL5 Standard Library .

| Class name                                   | Object                           |
|----------------------------------------------|----------------------------------|
| <a href="#"><u>CChartObjectText</u></a>      | Graphic object "Text"            |
| <a href="#"><u>CChartObjectLabel</u></a>     | Graphic object "Text Label"      |
| <a href="#"><u>CChartObjectEdit</u></a>      | Graphic object "Edit"            |
| <a href="#"><u>CChartObjectButton</u></a>    | Graphic object "Button"          |
| <a href="#"><u>CChartObjectSubChart</u></a>  | Graphic object "Chart"           |
| <a href="#"><u>CChartObjectBitmap</u></a>    | Graphic object "Bitmap"          |
| <a href="#"><u>CChartObjectBmpLabel</u></a>  | Graphic object "Bitmap Label"    |
| <a href="#"><u>CChartObjectRectLabel</u></a> | Graphic object "Rectangle Label" |

See also

[Object types](#), [Graphic objects](#)



## CChartObjectText

Class CChartObjectText is a class for simplified access to "Text" graphic object properties.

### Description

Class CChartObjectText provides access to "Text" object properties.

### Declaration

```
class CChartObjectText : public CChartObject
```

### Title

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

### Class Methods

| Create                       |                                      |
|------------------------------|--------------------------------------|
| <a href="#">Create</a>       | Creates graphic object "Text"        |
| Properties                   |                                      |
| <a href="#">Angle</a>        | Gets/Sets property "Angle"           |
| <a href="#">Font</a>         | Gets/Sets property "Font"            |
| <a href="#">FontSize</a>     | Gets/Sets property "FontSize"        |
| <a href="#">Anchor</a>       | Gets/Sets property "Anchor"          |
| Input/output                 |                                      |
| virtual <a href="#">Save</a> | Virtual method for writing to file   |
| virtual <a href="#">Load</a> | Virtual method for reading from file |
| virtual <a href="#">Type</a> | Virtual method of identification     |

Derived classes:

- [CChartObjectLabel](#)

See also

[Object types](#), [Object properties](#), [Methods of binding sites](#), [Graphic objects](#)

## Create

Creates graphic object "Text".

```
bool Create(  
    long      chart_id,    // Chart identifier  
    string     name,       // Object name  
    int       window,      // Chart window  
    datetime  time,        // Time coordinate  
    double    price        // Price coordinate  
)
```

### Parameters

*chart\_id*

[in] Chart identifier (0 - current chart).

*name*

[in] An unique name of the object to create.

*window*

[in] Chart window number (0 - base window).

*time*

[in] Time coordinate of the anchor point.

*price*

[in] Price coordinate of the anchor point.

### Returned value

true if successful, false if error.

## Angle (Get Method)

Gets the value of "Angle" property.

```
double Angle() const
```

### Returned value

Value of "Angle" property of the object, assigned to the class instance. If there is no object assigned, it returns EMPTY\_VALUE.

## Angle (Set Method)

Sets new value for "Angle" property.

```
bool Angle(  
    double angle    // new angle  
)
```

### Parameters

*angle*

[in] New value for "Angle" property.

### Returned value

true if successful, false if property hasn't changed.

## Font (Get Method)

Gets the value of "Font" property.

```
string Font() const
```

### Returned value

Value of "Font" property of the object, assigned to the class instance. If there is no object assigned, it returns "".

## Font (Set Method)

Sets new value for "Font" property.

```
bool Font(  
    string font    // new font  
)
```

### Parameters

*font*

[in] New value for "Font" property.

### Returned value

true if successful, false if property hasn't changed.

## FontSize (Get Method)

Gets the value of "FontSize" property.

```
int FontSize() const
```

### Returned value

Value of "FontSize" property of the object, assigned to the class instance. If there is no object assigned, it returns 0.

## FontSize (Set Method)

Sets new value for "FontSize" property.

```
bool FontSize(  
    int size    // new font size  
)
```

### Parameters

*size*

[in] New value for "Font" property.

### Returned value

true if successful, false if property hasn't changed.

## Anchor (Get Method)

Gets the value of "Anchor" property.

```
ENUM_ANCHOR_POINT Anchor() const
```

### Returned value

Value of "Anchor" property of the object, assigned to the class instance. If there is no object assigned, it returns WRONG\_VALUE.

## Anchor (Set Method)

Sets new value for "Anchor" property.

```
bool Anchor(  
    ENUM_ANCHOR_POINT anchor // new value  
)
```

### Parameters

*anchor*

[in] New value for "Anchor" property.

### Returned value

true if successful, false if property hasn't changed.

## Save

Saves object parameters to file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] handle of the binary file already opened by [FileOpen](#) function.

### Returned value

true if successful, false if error.

## Load

Loads object parameters from file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] handle of the binary file already opened by [FileOpen](#) function.

### Returned value

true if successful, false if error.



## Type

Returns graphic object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier (OBJ\_TEXT for [CChartObjectText](#)).

# CChartObjectLabel

Class CChartObjectLabel is a class for simplified access to "Label" graphic object properties.

## Description

Class CChartObjectLabel provides access to "Label" object properties.

## Declaration

```
class CChartObjectLabel : public CChartObjectText
```

## Title

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

## Class Methods

| Create                       |                                      |
|------------------------------|--------------------------------------|
| <a href="#">Create</a>       | Creates graphic object "Label"       |
| Properties                   |                                      |
| <a href="#">X_Distance</a>   | Gets/Sets property "X_Distance"      |
| <a href="#">Y_Distance</a>   | Gets/Sets property "Y_Distance"      |
| <a href="#">X_Size</a>       | Gets/Sets property "X_Size"          |
| <a href="#">Y_Size</a>       | Gets/Sets property "Y_Size"          |
| <a href="#">Corner</a>       | Gets/Sets property "Corner"          |
| <a href="#">Time</a>         | "Gag" for Time Coordinate change     |
| <a href="#">Price</a>        | "Gag" for Price Coordinate change    |
| Input/output                 |                                      |
| virtual <a href="#">Save</a> | Virtual method for writing to file   |
| virtual <a href="#">Load</a> | Virtual method for reading from file |
| virtual <a href="#">Type</a> | Virtual method of identification     |

Derived classes:

- [CChartObjectEdit](#)

See also

[Object types](#), [Object properties](#), [Chart angle](#), [Methods of binding sites](#), [Graphic objects](#)

## Create

Creates graphic object "Label".

```
bool Create(  
    long    chart_id,    // Chart identifier  
    string  name,        // Object name  
    int     window,      // Chart window  
    int     X,           // X coordinate  
    int     Y            // Y coordinate  
)
```

### Parameters

*chart\_id*

[in] Chart identifier (0 - current chart).

*name*

[in] An unique name of the object to create.

*window*

[in] Chart window number (0 - base window).

*X*

[in] X coordinate.

*Y*

[in] Y coordinate.

### Returned value

true if successful, false if error.

## X\_Distance (Get Method)

Gets the value of "X\_Distance" property.

```
int X_Distance() const
```

### Returned value

Value of "X\_Distance" property of the object, assigned to the class instance. If there is no object assigned, it returns 0.

## X\_Distance (Set Method)

Sets new value for "X\_Distance" property.

```
bool X_Distance(  
    int X // new value  
)
```

### Parameters

*X*

[in] New value for "X\_Distance" property.

### Returned value

true if successful, false if property hasn't changed.

## Y\_Distance (Get Method)

Gets the value of "Y\_Distance" property.

```
int Y_Distance() const
```

### Returned value

Value of "Y\_Distance" property of the object, assigned to the class instance. If there is no object assigned, it returns 0.

## Y\_Distance (Set Method)

Sets new value for "Y\_Distance" property.

```
bool Y_Distance(  
    int Y    // new value  
)
```

### Parameters

Y

[in] New value for "Y\_Distance" property.

### Returned value

true if successful, false if property hasn't changed.

## X\_Size

Gets the value of "X\_Size" property.

```
int X_Size() const
```

### Returned value

Value of "X\_Size" property of the object, assigned to the class instance. If there is no object assigned, it returns 0.

## Y\_Size

Gets the value of "Y\_Size" property.

```
int Y_Size() const
```

### Returned value

Value of "Y\_Size" property of the object, assigned to the class instance. If there is no object assigned, it returns 0.

## Corner (Get Method)

Gets the value of "Corner" property.

```
ENUM_BASE_CORNER Corner() const
```

### Returned value

Value of "Corner" property of the object, assigned to the class instance. If there is no object assigned, it returns WRONG\_VALUE.

## Corner (Set Method)

Sets new value for "Corner" property.

```
bool Corner(  
    ENUM_BASE_CORNER corner    // new value  
)
```

### Parameters

*corner*

[in] New value for "Corner" property.

### Returned value

true if successful, false if property hasn't changed.



## Time

Prohibits changes of the time coordinate.

```
bool Time(  
    datetime time    // any value  
)
```

### Parameters

*time*

[in] Any value of datetime type.

### Returned value

always false.

## Price

Prohibits changes of the price coordinate.

```
bool Price(  
    double price    // any value  
)
```

### Parameters

*price*

[in] Any value of double type.

### Returned value

always false.

## Save

Saves object parameters to file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] handle of the binary file already opened by [FileOpen](#) function.

### Returned value

true if successful, false if error.

## Load

Loads object parameters from file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] handle of the binary file already opened by [FileOpen](#) function.

### Returned value

true if successful, false if error.

## Type

Returns graphic object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier (OBJ\_LABEL for [CChartObjectLabel](#)).

## CChartObjectEdit

Class CChartObjectEdit is a class for simplified access to "Edit" graphic object properties.

### Description

Class CChartObjectEdit provides access to "Edit" object properties.

### Declaration

```
class CChartObjectEdit : public CChartObjectLabel
```

### Title

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

### Class Methods

| Create                       |                                       |
|------------------------------|---------------------------------------|
| <a href="#">Create</a>       | Creates "Edit" graphic object         |
| Properties                   |                                       |
| <a href="#">TextAlign</a>    | Gets/Sets "TextAlign" property        |
| <a href="#">X_Size</a>       | Gets "X Size" property                |
| <a href="#">Y_Size</a>       | Gets "Y Size" property                |
| <a href="#">BackColor</a>    | Gets/Sets "Background Color" property |
| <a href="#">BorderColor</a>  | Gets/Sets "Border Color" property     |
| <a href="#">ReadOnly</a>     | Gets/Sets "Read Only" property        |
| <a href="#">Angle</a>        | Gets/Sets "Angle" property            |
| Input/output                 |                                       |
| virtual <a href="#">Save</a> | Virtual method for writing to file    |
| virtual <a href="#">Load</a> | Virtual method for reading from file  |
| virtual <a href="#">Type</a> | Virtual method of identification      |

Derived classes:

- [CChartObjectButton](#)

See also

[Object types](#), [Object properties](#), [Chart angle](#), [Methods of binding sites](#), [Graphic objects](#)

## Create

Creates graphic object "Edit".

```
bool Create(  
    long    chart_id,    // Chart identifier  
    string  name,        // Object name  
    int     window,      // Chart window  
    int     X,           // X coordinate  
    int     Y,           // Y coordinate  
    int     sizeX,       // X size  
    int     sizeY        // Y size  
)
```

### Parameters

*chart\_id*

[in] Chart identifier (0 - current chart).

*name*

[in] An unique name of the object to create.

*window*

[in] Chart window number (0 - base window).

*X*

[in] X coordinate.

*Y*

[in] Y coordinate.

*sizeX*

[in] X size.

*sizeY*

[in] Y size.

### Returned value

true if successful, false if error.

## TextAlign (Get method)

Gets the value of "TextAlign" property ([text alignment mode](#)).

```
ENUM_ALIGN_MODE TextAlign() const
```

### Returned value

Value of "TextAlign" property of the object, assigned to the class instance.

## TextAlign (Set method)

Sets the value of "TextAlign" property ([text alignment mode](#)).

```
bool TextAlign(  
    ENUM_ALIGN_MODE align    // new value  
)
```

### Parameters

*align*

[in] New value of "TextAlign" property.

### Returned value

true if successful, false if property hasn't changed.



## X\_Size

Sets new value for "X\_Size" property.

```
bool X_Size(  
    int size    // new size  
)
```

### Parameters

*size*

[in] New value for "X\_Size" property.

### Returned value

true if successful, false if property hasn't changed.

## Y\_Size

Sets new value for "Y\_Size" property.

```
bool Y_Size(  
    int size    // new size  
)
```

### Parameters

*size*

[in] New value for "Y\_Size" property.

### Returned value

true if successful, false if property hasn't changed.

## BackColor (Get Method)

Gets the value of "BackColor" property.

```
color BackColor() const
```

### Returned value

Value of "BackColor" property of the object, assigned to the class instance. If there is no object assigned, it returns CLR\_NONE.

## BackColor (Set Method)

Sets new value for "BackColor" property.

```
bool BackColor(  
    color new_color    // new background color  
)
```

### Parameters

*new\_color*

[in] New value for "BackColor" property.

### Returned value

true if successful, false if property hasn't changed.

## BorderColor (Get Method)

Gets the value of "Border Color" property.

```
color BorderColor() const
```

### Returned value

Value of "Border Color" property of the object, assigned to the class instance. If there is no object assigned, it returns CLR\_NONE.

## BorderColor (Set Method)

Sets new value for "Border Color" property.

```
bool BorderColor (
    color new_color    // new border color
)
```

### Parameters

*new\_color*

[in] New value for "Border Color" property.

### Returned value

true if successful, false if property hasn't changed.

## ReadOnly (Get Method)

Gets the value of "Read Only" property.

```
bool ReadOnly() const
```

### Returned value

Value of "Read Only" property of the object, assigned to the class instance. If there is no object assigned, it returns false.

## ReadOnly (Set Method)

Sets new value for "Read Only" property.

```
bool ReadOnly(  
    const bool flag    // new value  
)
```

### Parameters

*flag*

[in] New value for "Read Only" property.

### Returned value

true if successful, false if property hasn't changed.

## Angle

Prohibits changes of the "Angle" property.

```
bool Angle(  
    double angle    // any value  
)
```

### Parameters

*angle*

[in] Any value of double type.

### Returned value

always false.

## Save

Saves object parameters to file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] handle of the binary file already opened by [FileOpen](#) function.

### Returned value

true if successful, false if error.

## Load

Loads object parameters from file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] handle of the binary file already opened by [FileOpen](#) function.

### Returned value

true if successful, false if error.



## Type

Returns graphic object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier (OBJ\_EDIT for [CChartObjectEdit](#)).

# CChartObjectButton

Class CChartObjectButton is a class for simplified access to "Button" graphic object properties.

## Description

Class CChartObjectButton provides access to "Button" object properties.

## Declaration

```
class CChartObjectButton : public CChartObjectEdit
```

## Title

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

## Class Methods

|                              |                                                       |
|------------------------------|-------------------------------------------------------|
| Create                       |                                                       |
| <a href="#">Create</a>       | Inherited from class <a href="#">CChartObjectEdit</a> |
| Properties                   |                                                       |
| <a href="#">State</a>        | Gets/Sets button state (Pressed/Depressed)            |
| Input/output                 |                                                       |
| virtual <a href="#">Save</a> | Virtual method for writing to file                    |
| virtual <a href="#">Load</a> | Virtual method for reading from file                  |
| virtual <a href="#">Type</a> | Virtual method of identification                      |

## See also

[Object types](#), [Object properties](#), [Chart angle](#), [Methods of binding sites](#), [Graphic objects](#)

## State (Get Method)

Gets the value of "State" property.

```
bool State() const
```

### Returned value

Value of "State" property of the object, assigned to the class instance. If there is no object assigned, it returns false.

## State (Set Method)

Sets new value for "State" property.

```
bool State(  
    bool state    // new state value  
)
```

### Parameters

*X*

[in] New value for "State" property.

### Returned value

true if successful, false if property hasn't changed.

## Save

Saves object parameters to file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] handle of the binary file already opened by [FileOpen](#) function.

### Returned value

true if successful, false if error.

## Load

Loads object parameters from file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] handle of the binary file already opened by [FileOpen](#) function.

### Returned value

true if successful, false if error.

## Type

Returns graphic object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier (OBJ\_BUTTON for [CChartObjectButton](#)).

## CChartObjectSubChart

Class CChartObjectSubChart is a class for simplified access to "Chart" graphic object properties.

### Description

Class CChartObjectSubChart provides access to "Chart" object properties.

### Declaration

```
class CChartObjectSubChart : public CChartObject
```

### Title

```
#include <ChartObjects\ChartObjectSubChart.mqh>
```

### Class Methods

| Create                       |                                       |
|------------------------------|---------------------------------------|
| <a href="#">Create</a>       | Creates graphic object "Chart"        |
| Properties                   |                                       |
| <a href="#">X_Distance</a>   | Gets/Sets property "X_Distance"       |
| <a href="#">Y_Distance</a>   | Gets/Sets property "Y_Distance"       |
| <a href="#">Corner</a>       | Gets/Sets property "Corner"           |
| <a href="#">X_Size</a>       | Gets/Sets property "X_Size"           |
| <a href="#">Y_Size</a>       | Gets/Sets property "Y_Size"           |
| <a href="#">Symbol</a>       | Gets/Sets property "Symbol"           |
| <a href="#">Period</a>       | Gets/Sets property "Period"           |
| <a href="#">Scale</a>        | Gets/Sets property "Scale"            |
| <a href="#">DateScale</a>    | Gets/Sets property "Show date scale"  |
| <a href="#">PriceScale</a>   | Gets/Sets property "Show price scale" |
| <a href="#">Time</a>         | "Gag" for time coordinate change      |
| <a href="#">Price</a>        | "Gag" for price coordinate change     |
| Input/output                 |                                       |
| virtual <a href="#">Save</a> | Virtual method for writing to file    |
| virtual <a href="#">Load</a> | Virtual method for reading from file  |
| virtual <a href="#">Type</a> | Virtual method of identification      |

### See also

[Object types](#), [Object properties](#), [Chart angle](#), [Graphic objects](#)



## Create

Creates graphic object "SubChart".

```
bool Create(  
    long    chart_id,    // Chart identifier  
    string  name,        // Object name  
    int     window,      // Chart window  
    int     X,           // X coordinate  
    int     Y,           // Y coordinate  
    int     sizeX,       // X size  
    int     sizeY        // Y size  
)
```

### Parameters

*chart\_id*

[in] Chart identifier (0 - current chart).

*name*

[in] An unique name of the object to create.

*window*

[in] Chart window number (0 - base window).

*X*

[in] X coordinate.

*Y*

[in] Y coordinate.

*sizeX*

[in] X size.

*sizeY*

[in] Y size.

### Returned value

true if successful, false if error.

## X\_Distance (Get Method)

Gets the value of "X\_Distance" property.

```
int X_Distance() const
```

### Returned value

Value of "X\_Distance" property of the object, assigned to the class instance. If there is no object assigned, it returns 0.

## X\_Distance (Set Method)

Sets new value for "X\_Distance" property.

```
bool X_Distance(  
    int X      // new value  
)
```

### Parameters

*X*

[in] New value for "X\_Distance" property.

### Returned value

true if successful, false if property hasn't changed.

## Y\_Distance (Get Method)

Gets the value of "Y\_Distance" property.

```
int Y_Distance() const
```

### Returned value

Value of "Y\_Distance" property of the object, assigned to the class instance. If there is no object assigned, it returns 0.

## Y\_Distance (Set Method)

Sets new value for "Y\_Distance" property.

```
bool Y_Distance(  
    int Y    // new value  
)
```

### Parameters

*Y*

[in] New value for "Y\_Distance" property.

### Returned value

true if successful, false if property hasn't changed.

## Corner (Get Method)

Gets the value of "Corner" property.

```
ENUM_BASE_CORNER Corner() const
```

### Returned value

Value of "Corner" property of the object, assigned to the class instance. If there is no object assigned, it returns WRONG\_VALUE.

## Corner (Set Method)

Sets new value for "Corner" property.

```
bool Corner(  
    ENUM_BASE_CORNER corner    // new value  
)
```

### Parameters

*corner*

[in] New value for "Corner" property.

### Returned value

true if successful, false if property hasn't changed.

## X\_Size (Get Method)

Gets the value of "X\_Size" property.

```
int X_Size() const
```

### Returned value

Value of "X\_Size" property of the object, assigned to the class instance. If there is no object assigned, it returns 0.

## X\_Size (Set Method)

Sets new value for "X\_Size" property.

```
bool X_Size(  
    int X    // new value  
)
```

### Parameters

X

[in] New value for "X\_Size" property.

### Returned value

true if successful, false if property hasn't changed.

## Y\_Size (Get Method)

Gets the value of "Y\_Size" property.

```
int Y_Size() const
```

### Returned value

Value of "Y\_Size" property of the object, assigned to the class instance. If there is no object assigned, it returns 0.

## Y\_Size (Set Method)

Sets new value for "Y\_Size" property.

```
bool Y_Size(  
    int Y    // new value  
)
```

### Parameters

Y

[in] New value for "Y\_Size" property.

### Returned value

true if successful, false if property hasn't changed.

## Symbol (Get Method)

Gets the value of "Symbol" property.

```
string Symbol() const
```

### Returned value

Value of "Symbol" property of the object, assigned to the class instance. If there is no object assigned, it returns "".

## Symbol (Set Method)

Sets new value for "Symbol" property.

```
bool Symbol(  
    string symbol    // new symbol  
)
```

### Parameters

*symbol*

[in] New value for "Symbol" property.

### Returned value

true if successful, false if property hasn't changed.

## Period (Get Method)

Gets the value of "Period" property.

```
int Period() const
```

### Returned value

Value of "Period" property of the object, assigned to the class instance. If there is no object assigned, it returns 0.

## Period (Set Method)

Sets new value for "Period" property.

```
bool Period(  
    int period    // new period  
)
```

### Parameters

*period*

[in] New value for "Period" property.

### Returned value

true if successful, false if property hasn't changed.



## Scale (Get Method)

Gets the value of "Scale" property.

```
double Scale() const
```

### Returned value

Value of "Scale" property of the object, assigned to the class instance. If there is no object assigned, it returns EMPTY\_VALUE.

## Scale (Set Method)

Sets new value for "Scale" property.

```
bool Scale(  
    double scale    // new scale  
)
```

### Parameters

*scale*

[in] New value for "Scale" property.

### Returned value

true if successful, false if property hasn't changed.

## DateScale (Get Method)

Gets the value of "DateScale" property.

```
bool DateScale() const
```

### Returned value

Value of "DateScale" property of the object, assigned to the class instance. If there is no object assigned, it returns false.

## DateScale (Set Method)

Sets new value for "DateScale" property.

```
bool DateScale (  
    bool scale    // new value  
)
```

### Parameters

*scale*

[in] New value for "DateScale" property.

### Returned value

true if successful, false if property hasn't changed.

## PriceScale (Get Method)

Gets the value of "PriceScale" property.

```
bool PriceScale() const
```

### Returned value

Value of "PriceScale" property of the object, assigned to the class instance. If there is no object assigned, it returns false.

## PriceScale (Set Method)

Sets new value for "PriceScale" property.

```
bool PriceScale(  
    bool scale    // new value  
)
```

### Parameters

*scale*

[in] New value for "PriceScale" property.

### Returned value

true if successful, false if property hasn't changed.

## Time

Prohibits changes of the time coordinate.

```
bool Time(  
    datetime time    // any value  
)
```

### Parameters

*time*

[in] Any value of datetime type.

### Returned value

always false.

## Price

Prohibits changes of the price coordinate.

```
bool Price(  
    double price    // any value  
)
```

### Parameters

*price*

[in] Any value of double type.

### Returned value

always false.

## Save

Saves object parameters to file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] handle of the binary file already opened by [FileOpen](#) function.

### Returned value

true if successful, false if error.

## Load

Loads object parameters from file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] handle of the binary file already opened by [FileOpen](#) function.

### Returned value

true if successful, false if error.

## Type

Returns graphic object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier (OBJ\_CHART for [CChartObjectSubChart](#)).



# CChartObjectBitmap

Class CChartObjectBitmap is a class for simplified access to "Bitmap" graphic object properties.

## Description

Class CChartObjectBitmap provides access to "Bitmap" object properties.

## Declaration

```
class CChartObjectBitmap : public CChartObject
```

## Title

```
#include <ChartObjects\ChartObjectsBmpControls.mqh>
```

## Class Methods

|                              |                                      |
|------------------------------|--------------------------------------|
| Create                       |                                      |
| <a href="#">Create</a>       | Creates graphic object "Bitmap"      |
| Properties                   |                                      |
| <a href="#">BmpFile</a>      | Gets/Sets property "BMP Filename"    |
| <a href="#">X_Offset</a>     | Gets/Sets property "X_Offset"        |
| <a href="#">Y_Offset</a>     | Gets/Sets property "Y_Offset"        |
| Input/output                 |                                      |
| virtual <a href="#">Save</a> | Virtual method for writing to file   |
| virtual <a href="#">Load</a> | Virtual method for reading from file |
| virtual <a href="#">Type</a> | Virtual method of identification     |

### See also

[Object types](#), [Object properties](#), [Graphic objects](#)

## Create

Creates graphic object "Bitmap".

```
bool Create(  
    long      chart_id,    // Chart identifier  
    string     name,       // Object name  
    int       window,      // Chart window  
    datetime   time,       // Time coordinate  
    double     price       // Price coordinate  
)
```

### Parameters

*chart\_id*

[in] Chart identifier (0 - current chart).

*name*

[in] An unique name of the object to create.

*window*

[in] Chart window number (0 - base window).

*time*

[in] Time coordinate of the anchor point.

*price*

[in] Price coordinate of the anchor point.

### Returned value

true if successful, false if error.

## BmpFile (Get Method)

Gets the value of "BmpFile" property.

```
string BmpFile() const
```

### Returned value

Value of "BmpFile" property of the object, assigned to the class instance. If there is no object assigned, it returns false.

## BmpFile (Set Method)

Sets new value for "BmpFile" property.

```
bool BmpFile(  
    string name    // new file name  
)
```

### Parameters

X

[in] New value for "BmpFile" property.

### Returned value

true if successful, false if property hasn't changed.

## X\_Offset (Get Method)

Gets the value of "X\_Offset" property (the X coordinate of the upper left corner of the rectangular visible area in the graphical objects).

```
int X_Offset() const
```

### Returned value

Value of "X\_Offset" property of the object, assigned to the class instance. If there is no object assigned, it returns 0.

## X\_Offset (Set Method)

Sets new value for "X\_Offset" property (the X coordinate of the upper left corner of the rectangular visible area in the graphical objects). The value is set in pixels relative to the upper left corner of the original image.

```
bool X_Offset(  
    int X    // new value  
)
```

### Parameters

X

[in] New value for "X\_Offset" property.

### Returned value

true if successful, false if property hasn't changed.

## Y\_Offset (Get Method)

Gets the value of "Y\_Offset" property (the Y coordinate of the upper left corner of the rectangular visible area in the graphical objects).

```
int Y_Offset() const
```

### Returned value

Value of "Y\_Offset" property of the object, assigned to the class instance. If there is no object assigned, it returns 0.

## Y\_Offset (Set Method)

Sets new value for "Y\_Offset" property (the Y coordinate of the upper left corner of the rectangular visible area in the graphical objects). The value is set in pixels relative to the upper left corner of the original image.

```
bool Y_Offset(  
    int Y    // new value  
)
```

### Parameters

Y

[in] New value for "Y\_Offset" property.

### Returned value

true if successful, false if property hasn't changed.

## Save

Saves object parameters to file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] handle of the binary file already opened by [FileOpen](#) function.

### Returned value

true if successful, false if error.

## Load

Loads object parameters from file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] handle of the binary file already opened by [FileOpen](#) function.

### Returned value

true if successful, false if error.

## Type

Returns graphic object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier (OBJ\_BITMAP for [CChartObjectBitmap](#)).



## CChartObjectBmpLabel

Class CChartObjectBmpLabel is a class for simplified access to "Bitmap Label" graphic object properties.

### Description

Class CChartObjectBmpLabel provides access to "Bitmap Label" object properties.

### Declaration

```
class CChartObjectBmpLabel : public CChartObject
```

### Title

```
#include <ChartObjects\ChartObjectsBmpControls.mqh>
```

### Class Methods

| Create                       |                                                                  |
|------------------------------|------------------------------------------------------------------|
| <a href="#">Create</a>       | Creates graphic object "BmpLabel"                                |
| Properties                   |                                                                  |
| <a href="#">X_Distance</a>   | Gets/Sets property "X_Distance"                                  |
| <a href="#">Y_Distance</a>   | Gets/Sets property "Y_Distance"                                  |
| <a href="#">X_Offset</a>     | Gets/Sets property "X_Offset"                                    |
| <a href="#">Y_Offset</a>     | Gets/Sets property "Y_Offset"                                    |
| <a href="#">Corner</a>       | Gets/Sets property "Corner"                                      |
| <a href="#">X_Size</a>       | Gets/Sets property "X_Size"                                      |
| <a href="#">Y_Size</a>       | Gets/Sets property "Y_Size"                                      |
| <a href="#">BmpFileOn</a>    | Gets/Sets property "BmpFileOn" for button pressed state (On)     |
| <a href="#">BmpFileOff</a>   | Gets/Sets property "BmpFileOff" for button depressed state (Off) |
| <a href="#">State</a>        | Gets/Sets property "Button State" (Pressed/Depressed)            |
| <a href="#">Time</a>         | "Gag" for time coordinate change                                 |
| <a href="#">Price</a>        | "Gag" for price coordinate change                                |
| Input/output                 |                                                                  |
| virtual <a href="#">Save</a> | Virtual method for writing to file                               |
| virtual <a href="#">Load</a> | Virtual method for reading from file                             |

|                              |                                  |
|------------------------------|----------------------------------|
| virtual <a href="#">Type</a> | Virtual method of identification |
|------------------------------|----------------------------------|

**See also**

[Object types](#), [Object properties](#), [Chart angle](#), [Graphic objects](#)

## Create

Creates graphic object "BmpLabel".

```
bool Create(  
    long    chart_id,    // Chart identifier  
    string  name,        // Object name  
    int     window,      // Chart window  
    int     X,           // X coordinate  
    int     Y            // Y coordinate  
)
```

### Parameters

*chart\_id*

[in] Chart identifier (0 - current chart).

*name*

[in] A unique name of the object to create.

*window*

[in] Chart window number (0 - base window).

*X*

[in] X coordinate.

*Y*

[in] Y coordinate.

### Returned value

true if successful, false if error.

## X\_Distance (Get Method)

Gets the value of "X\_Distance" property.

```
int X_Distance() const
```

### Returned value

Value of "X\_Distance" property of the object, assigned to the class instance. If there is no object assigned, it returns 0.

## X\_Distance (Set Method)

Sets new value for "X\_Distance" property.

```
bool X_Distance(  
    int X    // new value  
)
```

### Parameters

*X*

[in] New value for "X\_Distance" property.

### Returned value

true if successful, false if property hasn't changed.

## Y\_Distance (Get Method)

Gets the value of "Y\_Distance" property.

```
int Y_Distance() const
```

### Returned value

Value of "Y\_Distance" property of the object, assigned to the class instance. If there is no object assigned, it returns 0.

## Y\_Distance (Set Method)

Sets new value for "Y\_Distance" property.

```
bool Y_Distance(  
    int Y // new value  
)
```

### Parameters

Y

[in] New value for "Y\_Distance" property.

### Returned value

true if successful, false if property hasn't changed.

## X\_Offset (Get Method)

Gets the value of "X\_Offset" property (the X coordinate of the upper left corner of the rectangular visible area in the graphical objects).

```
int X_Offset() const
```

### Returned value

Value of "X\_Offset" property of the object, assigned to the class instance. If there is no object assigned, it returns 0.

## X\_Offset (Set Method)

Sets new value for "X\_Offset" property (the X coordinate of the upper left corner of the rectangular visible area in the graphical objects). The value is set in pixels relative to the upper left corner of the original image.

```
bool X_Offset(  
    int X    // new value  
)
```

### Parameters

X

[in] New value for "X\_Offset" property.

### Returned value

true if successful, false if property hasn't changed.

## Y\_Offset (Get Method)

Gets the value of "Y\_Offset" property (the Y coordinate of the upper left corner of the rectangular visible area in the graphical objects).

```
int Y_Offset() const
```

### Returned value

Value of "Y\_Offset" property of the object, assigned to the class instance. If there is no object assigned, it returns 0.

## Y\_Offset (Set Method)

Sets new value for "Y\_Offset" property (the Y coordinate of the upper left corner of the rectangular visible area in the graphical objects). The value is set in pixels relative to the upper left corner of the original image.

```
bool Y_Offset(  
    int Y    // new value  
)
```

### Parameters

Y

[in] New value for "Y\_Offset" property.

### Returned value

true if successful, false if property hasn't changed.

## Corner (Get Method)

Gets the value of "Corner" property.

```
ENUM_BASE_CORNER Corner() const
```

### Returned value

Value of "Corner" property of the object, assigned to the class instance. If there is no object assigned, it returns WRONG\_VALUE.

## Corner (Set Method)

Sets new value for "Corner" property.

```
bool Corner(  
    ENUM_BASE_CORNER corner    // new value  
)
```

### Parameters

*corner*

[in] New value for "Corner" property.

### Returned value

true if successful, false if property hasn't changed.



## X\_Size

Gets the value of "X\_Size" property.

```
int X_Size() const
```

### Returned value

Value of "X\_Size" property of the object, assigned to the class instance. If there is no object assigned, it returns 0.

## Y\_Size

Gets the value of "Y\_Size" property.

```
int Y_Size() const
```

### Returned value

Value of "Y\_Size" property of the object, assigned to the class instance. If there is no object assigned, it returns 0.

## BmpFileOn (Get Method)

Gets the value of "BmpFileOn" property.

```
string BmpFileOn() const
```

### Returned value

Value of "BmpFileOn" property of the object, assigned to the class instance. If there is no object assigned, it returns "".

## BmpFileOn (Set Method)

Sets new value for "BmpFileOn" property.

```
bool BmpFileOn (
    string name      // file name
)
```

### Parameters

*name*

[in] New value for "BmpFileOn" property.

### Returned value

true if successful, false if property hasn't changed.

## BmpFileOff (Get Method)

Gets the value of "BmpFileOff" property.

```
string BmpFileOff() const
```

### Returned value

Value of "BmpFileOff" property of the object, assigned to the class instance. If there is no object assigned, it returns "".

## BmpFileOff (Set Method)

Sets new value for "BmpFileOff" property.

```
bool BmpFileOff(  
    string name    // file name  
)
```

### Parameters

*name*

[in] New value for "BmpFileOff" property.

### Returned value

true if successful, false if property hasn't changed.

## State (Get Method)

Gets the value of "State" property.

```
bool State() const
```

### Returned value

Value of "State" property of the object, assigned to the class instance. If there is no object assigned, it returns false.

## State (Set Method)

Sets new value for "State" property.

```
bool State(  
    bool state    // new state value  
)
```

### Parameters

X

[in] New value for "State" property.

### Returned value

true if successful, false if property hasn't changed.

## Time

Prohibits changes of the time coordinate.

```
bool Time(  
    datetime time    // any value  
)
```

### Parameters

*time*

[in] Any value of datetime type.

### Returned value

always false.

## Price

Prohibits changes of the price coordinate.

```
bool Price(  
    double price    // any value  
)
```

### Parameters

*price*

[in] Any value of double type.

### Returned value

always false.

## Save

Saves object parameters to file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] handle of the binary file already opened by [FileOpen](#) function.

### Returned value

true if successful, false if error.



## Load

Loads object parameters from file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] handle of the binary file already opened by [FileOpen](#) function.

### Returned value

true if successful, false if error.

## Type

Returns graphic object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier (OBJ\_BITMAP\_LABEL for [CChartObjectBmpLabel](#)).

## CChartObjectRectLabel

Class CChartObjectRectLabel is a class for simplified access to "Rectangle Label" graphic object properties.

### Description

Class CChartObjectRectLabel provides access to "Rectangle Label" object properties.

### Declaration

```
class CChartObjectRectLabel : public CChartObjectLabel
```

### Title

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

### Class Methods

| Create                       |                                      |
|------------------------------|--------------------------------------|
| <a href="#">Create</a>       | Creates "RectLabel" graphic object   |
| Properties                   |                                      |
| <a href="#">X_Size</a>       | Sets the horizontal size             |
| <a href="#">Y_Size</a>       | Sets the vertical size               |
| <a href="#">BackColor</a>    | Gets/Sets the background color       |
| <a href="#">Angle</a>        | A gag method                         |
| <a href="#">BorderType</a>   | Gets/Sets type of the border         |
| Input/output                 |                                      |
| virtual <a href="#">Save</a> | Virtual method for writing to file   |
| virtual <a href="#">Load</a> | Virtual method for reading from file |
| virtual <a href="#">Type</a> | Virtual method of identification     |

### See also

[Object types](#), [Object properties](#), [Graphic objects](#)

## Create

Creates the "CChartObjectRectLabel" graphic object.

```
bool Create(  
    long    chart_id,    // Chart ID  
    string  name,        // Object name  
    int     window,      // Chart window  
    int     X,           // X coordinate  
    int     Y,           // Y coordinate  
    int     sizeX,       // Horizontal size  
    int     sizeY        // Vertical size  
)
```

### Parameters

*chart\_id*

[in] Chart identifier (0 - current chart).

*name*

[in] A unique name of the object to create.

*window*

[in] Chart window number (0 - base window).

*X*

[in] X coordinate.

*Y*

[in] Y coordinate.

*sizeX*

[in] Horizontal size.

*sizeY*

[in] Vertical size.

### Returned value

true if successful, false if error.

## X\_Size

Sets the value of "X\_Size" property.

```
bool X_Size(  
    int size    // Horizontal size  
)
```

### Parameters

*size*

[in] New horizontal size.

### Returned value

true if successful, false if the property hasn't been changed.

### Note

To get the values of "X\_Size" and "Y\_Size" properties use the [X\\_Size](#) and [Y\\_Size](#) methods of the parent [CChartObjectLabel](#) class.

## Y\_Size

Sets the value of "Y\_Size" property.

```
bool Y_Size(  
    int size    // Vertical size  
)
```

### Parameters

*size*

[in] New vertical size.

### Returned value

true if successful, false if the property hasn't been changed.

### Note

To get the values of "X\_Size" and "Y\_Size" properties use the [X\\_Size](#) and [Y\\_Size](#) methods of the parent [CChartObjectLabel](#) class.

## BackColor

Gets the background color.

```
color BackColor() const
```

### Returned value

Background color of the object, assigned to the class instance. If there is no object assigned, it returns 0.

## BackColor

Sets the background color.

```
bool BackColor(  
    color new_color    // New color  
)
```

### Parameters

*new\_color*

[in] New background color.

### Returned value

true if successful, false if property hasn't been changed.

## Angle

A gag method.

```
bool Angle(  
    double angle    // any value  
)
```

### Parameters

*angle*

[in] Any value of [double](#) type.

### Returned value

Always false.



## BorderType

Gets border type.

```
int BorderType() const
```

### Returned value

Border type of the object, assigned to the class instance. If there is no object assigned, it returns 0.

## BorderType

Sets border type.

```
bool BorderType(  
    int type    // Border type  
)
```

### Parameters

*type*

[in] New border type.

### Returned value

true if successful, false if property hasn't been changed.

## Save

Saves object parameters to file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] handle of the binary file already opened by [FileOpen](#) function.

### Returned value

true if successful, false if error.

## Load

Loads object parameters from file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] handle of the binary file already opened by [FileOpen](#) function.

### Returned value

true if successful, false if error.

## Type

Returns graphic object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier (OBJ\_RECTANGLE\_LABEL for [CChartObjectRectangleLabel](#)).

## CCanvas

CCanvas is a class for simplified creation of custom images.

### Description

CCanvas provides creation of a graphical resource (with or without binding to a chart object) and drawing graphic primitives.

### Declaration

```
class CCanvas
```

### Title

```
#include <Canvas\Canvas.mqh>
```

### Class Methods by Groups

| Creating                          |                                                                |
|-----------------------------------|----------------------------------------------------------------|
| <a href="#">Create</a>            | Creates a graphical resource without binding to a chart object |
| <a href="#">CreateBitmap</a>      | Creates a graphical resource bound to a chart object           |
| <a href="#">CreateBitmapLabel</a> | Creates a graphical resource bound to a chart object           |
| <a href="#">Destroy</a>           | Destroys a graphical resource                                  |
| Properties                        |                                                                |
| <a href="#">ChartObjectName</a>   | Receives the name of a bound chart object                      |
| <a href="#">ResourceName</a>      | Receives the name of a graphical resource                      |
| <a href="#">Width</a>             | Receives the width of a graphical resource                     |
| <a href="#">Height</a>            | Receives the height of a graphical resource                    |
| <a href="#">LineStyleSet</a>      | Sets the line style                                            |
| Updating an object on the screen  |                                                                |
| <a href="#">Update</a>            | Displays changes on the screen                                 |
| <a href="#">Resize</a>            | Resizes a graphical resource                                   |
| Erasing/Filling with color        |                                                                |
| <a href="#">Erase</a>             | Erases or fills with the specified color                       |
| Data access                       |                                                                |
| <a href="#">PixelGet</a>          | Receives color of the dot with the specified coordinates       |

|                                             |                                                      |
|---------------------------------------------|------------------------------------------------------|
| <a href="#"><u>PixelSet</u></a>             | Sets color of the dot with the specified coordinates |
| <b>Drawing primitives</b>                   |                                                      |
| <a href="#"><u>LineVertical</u></a>         | Draws a vertical line                                |
| <a href="#"><u>LineHorizontal</u></a>       | Draws a horizontal line                              |
| <a href="#"><u>Line</u></a>                 | Draws a freehand line                                |
| <a href="#"><u>Polyline</u></a>             | Draws a polyline                                     |
| <a href="#"><u>Polygon</u></a>              | Draws a polygon                                      |
| <a href="#"><u>Rectangle</u></a>            | Draws a rectangle                                    |
| <a href="#"><u>Circle</u></a>               | Draws a circle                                       |
| <a href="#"><u>Triangle</u></a>             | Draws a triangle                                     |
| <b>Drawing filled primitives</b>            |                                                      |
| <a href="#"><u>FillRectangle</u></a>        | Draws a filled rectangle                             |
| <a href="#"><u>FillCircle</u></a>           | Draws a filled circle                                |
| <a href="#"><u>FillTriangle</u></a>         | Draws a filled triangle                              |
| <a href="#"><u>Fill</u></a>                 | Fills an area                                        |
| <b>Drawing primitives with antialiasing</b> |                                                      |
| <a href="#"><u>PixelSetAA</u></a>           | Draws a pixel                                        |
| <a href="#"><u>LineAA</u></a>               | Draws a line                                         |
| <a href="#"><u>PolylineAA</u></a>           | Draws a polyline                                     |
| <a href="#"><u>PolygonAA</u></a>            | Draws a polygon                                      |
| <a href="#"><u>TriangleAA</u></a>           | Draws a triangle                                     |
| <a href="#"><u>CircleAA</u></a>             | Draws a circle                                       |
| <b>Text</b>                                 |                                                      |
| <a href="#"><u>FontSet</u></a>              | Sets font parameters                                 |
| <a href="#"><u>FontNameSet</u></a>          | Sets font name                                       |
| <a href="#"><u>FontSizeSet</u></a>          | Sets font size                                       |
| <a href="#"><u>FontFlagsSet</u></a>         | Sets font flags                                      |
| <a href="#"><u>FontAngleSet</u></a>         | Sets font slope angle                                |
| <a href="#"><u>FontGet</u></a>              | Receives font parameters                             |
| <a href="#"><u>FontNameGet</u></a>          | Receives font name                                   |
| <a href="#"><u>FontSizeGet</u></a>          | Receives font size                                   |
| <a href="#"><u>FontFlagsGet</u></a>         | Receives font flags                                  |

|                                     |                                |
|-------------------------------------|--------------------------------|
| <a href="#">FontAngleGet</a>        | Receives font slope angle      |
| <a href="#">TextOut</a>             | Displays text                  |
| <a href="#">TextWidth</a>           | Receives the text width        |
| <a href="#">TextHeight</a>          | Receives the text height       |
| <a href="#">TextSize</a>            | Receives the text size         |
| <b>Transparency</b>                 |                                |
| <a href="#">TransparentLevelSet</a> | Sets transparency level        |
| <b>Input/output</b>                 |                                |
| <a href="#">LoadFromFile</a>        | Reads an image from a BMP file |

## ChartObjectName

Receives the name of a bound chart object.

```
string ChartObjectName();
```

### Returned value

the name of a bound chart object



## Circle

Draws a circle

```
void Circle(  
    int      x,      // X coordinate  
    int      y,      // Y coordinate  
    int      r,      // radius  
    const uint clr    // color  
);
```

### Parameters

*x*

[in] X coordinate of the center of the circle.

*y*

[in] Y coordinate of the center of the circle.

*r*

[in] Circle radius.

*clr*

[in] Color in ARGB format.

## CircleAA

Draws a circle using antialiasing algorithm

```
void CircleAA(  
    const int    x,        // X coordinate  
    const int    y,        // Y coordinate  
    const double r,        // radius  
    const uint   clr       // color  
) ;
```

### Parameters

*x*

[in] X coordinate of the center of the circle.

*y*

[in] Y coordinate of the center of the circle.

*r*

[in] Circle radius.

*clr*

[in] Color in ARGB format.

## Create

Creates a graphical resource without binding to a chart object.

```
virtual bool Create(  
    const string      name,                // name  
    const int         width,              // width  
    const int         height,            // height  
    ENUM_COLOR_FORMAT clrfmt=COLOR_FORMAT_XRGB_NOALPHA // format  
);
```

### Parameters

*name*

[in] Basis for a graphical resource name. A resource name is generated during the creation by adding a pseudorandom string.

*width*

[in] Width (size along X axis) in pixels.

*height*

[in] Height (size along Y axis) in pixels.

*clrfmt=COLOR\_FORMAT\_XRGB\_NOALPHA*

[in] Color processing method. See [ResourceCreate\(\)](#) function description to learn more about color processing methods.

### Returned value

true - if successful, otherwise - false

## CreateBitmap

Creates a graphical resource bound to a chart object.

1. Creates a graphical resource in the main window of the current chart.

```
bool CreateBitmap(
    const string      name,                // name
    const datetime    time,               // time
    const double      price,              // price
    const int         width,              // width
    const int         height,             // height
    ENUM_COLOR_FORMAT clrfmt=COLOR_FORMAT_XRGB_NOALPHA // format
);
```

2. Creates a graphical resource using a chart ID and a subwindow number.

```
bool CreateBitmap(
    const long        chart_id,           // chart ID
    const int         subwin,             // subwindow number
    const string      name,              // name
    const datetime    time,              // time
    const double      price,             // price
    const int         width,             // width
    const int         height,            // height
    ENUM_COLOR_FORMAT clrfmt=COLOR_FORMAT_XRGB_NOALPHA // format
);
```

### Parameters

*chart\_id*

[in] Chart ID for creating an object.

*subwin*

[in] Chart subwindow number for creating an object.

*name*

[in] Chart object name and a basis for a graphical resource name.

*time*

[in] Chart object anchor point time coordinate.

*price*

[in] Chart object anchor point price coordinate.

*width*

[in] Graphical resource width (size along X axis) in pixels.

*height*

[in] Graphical resource height (size along Y axis) in pixels.

*clrfmt=COLOR\_FORMAT\_XRGB\_NOALPHA*

[in] Color processing method. See [ResourceCreate\(\)](#) function description to learn more about color processing methods.

**Returned value**

true - if successful, otherwise - false

**Note**

If the first function version is used, the object is created in the main window of the current chart.

Object size coincides with the size of a graphical resource.

## CreateBitmapLabel

Creates a graphical resource bound to a chart object.

1. Creates a graphical resource in the main window of the current chart.

```
bool CreateBitmapLabel(
    const string      name,                // name
    const int         x,                   // X coordinate
    const int         y,                   // Y coordinate
    const int         width,               // width
    const int         height,              // height
    ENUM_COLOR_FORMAT clrfmt=COLOR_FORMAT_XRGB_NOALPHA // format
);
```

2. Creates a graphical resource using a chart ID and a subwindow number.

```
bool CreateBitmapLabel(
    const long        chart_id,            // chart ID
    const int         subwin,              // subwindow number
    const string      name,                // name
    const int         x,                   // X coordinate
    const int         y,                   // Y coordinate
    const int         width,               // width
    const int         height,              // height
    ENUM_COLOR_FORMAT clrfmt=COLOR_FORMAT_XRGB_NOALPHA // format
);
```

### Parameters

*chart\_id*

[in] Chart ID for creating an object.

*subwin*

[in] Chart subwindow number for creating an object.

*name*

[in] Chart object name and a basis for a graphical resource name.

*x*

[in] Chart object anchor point X coordinate.

*y*

[in] Chart object anchor point Y coordinate.

*width*

[in] Graphical resource width (size along X axis) in pixels.

*height*

[in] Graphical resource height (size along Y axis) in pixels.

*clrfmt=COLOR\_FORMAT\_XRGB\_NOALPHA*

[in] Color processing method. See [ResourceCreate\(\)](#) function description to learn more about color processing methods.

**Returned value**

true - if successful, otherwise - false

**Note**

If the first function version is used, the object is created in the main window of the current chart.

Object size coincides with the size of a graphical resource.

## Destroy

Destroys a graphical resource.

```
void Destroy();
```

### Note

If a graphical resource has been bound to a chart object, the latter is deleted.



## Erase

Erases or fills with the specified color.

```
void Erase(  
    const uint clr=0    // color  
);
```

### Parameters

*clr=0*

[in] Color in ARGB format.

## Fill

Fills an area.

```
void Fill(  
    int      x,          // X coordinate  
    int      y,          // Y coordinate  
    const uint clr       // color  
);
```

### Parameters

*x*

[in] X coordinate of filling starting point.

*y*

[in] Y coordinate of filling starting point.

*clr*

[in] Color in ARGB format.

## FillCircle

Draws a filled circle.

```
void FillCircle(  
    int      x,      // X coordinate  
    int      y,      // Y coordinate  
    int      r,      // radius  
    const uint clr    // color  
);
```

### Parameters

*x*

[in] X coordinate of a filled circle center.

*y*

[in] Y coordinate of a filled circle center.

*r*

[in] Filled circle radius.

*clr*

[in] Color in ARGB format.

## FillRectangle

Draws a filled rectangle.

```
void FillRectangle(  
    int      x1,      // X coordinate  
    int      y1,      // Y coordinate  
    int      x2,      // X coordinate  
    int      y2,      // Y coordinate  
    const uint clr     // color  
);
```

### Parameters

*x1*

[in] X coordinate of the first point forming a rectangle.

*y1*

[in] Y coordinate of the first point forming a rectangle.

*x2*

[in] X coordinate of the second point forming a rectangle.

*y2*

[in] Y coordinate of the second point forming a rectangle.

*clr*

[in] Color in ARGB format.

## FillTriangle

Draws a filled triangle.

```
void FillTriangle(  
    int      x1,      // X coordinate  
    int      y1,      // Y coordinate  
    int      x2,      // X coordinate  
    int      y2,      // Y coordinate  
    int      x3,      // X coordinate  
    int      y3,      // Y coordinate  
    const uint clr     // color  
);
```

### Parameters

*x1*

[in] X coordinate of the triangle's first corner.

*y1*

[in] Y coordinate of the triangle's first corner.

*x2*

[in] X coordinate of the triangle's second corner.

*y2*

[in] Y coordinate of the triangle's second corner.

*x3*

[in] X coordinate of the triangle's third corner.

*y3*

[in] Y coordinate of the triangle's third corner.

*clr*

[in] Color in ARGB format.

## FontAngleGet

Receives font slope angle.

```
uint FontAngleGet ();
```

### Returned value

font slope angle

## FontAngleSet

Sets font slope angle.

```
bool FontAngleSet(  
    uint angle    // angle  
);
```

### Parameters

*angle*

[in] Font slope angle in tenths of a degree.

### Returned value

true - if successful, otherwise - false

## FontFlagsGet

Receives font flags.

```
uint FontFlagsGet ();
```

### Returned value

font flags



## FontFlagsSet

Sets font flags.

```
bool FontFlagsSet(  
    uint flags    // flags  
);
```

### Parameters

*flags*

[in] Font creation flags. See [TextSetFont\(\)](#) function description to learn more about the flags.

### Returned value

true - if successful, otherwise - false

## FontGet

Receives the current font parameters.

```
void FontGet(  
    string& name,      // name  
    int& size,         // size  
    uint& flags,       // flags  
    uint& angle        // slope angle  
);
```

### Parameters

*name*

[out] Reference to the variable for returning a font name.

*size*

[out] Reference to the variable for returning a font size.

*flags*

[out] Reference to the variable for returning font flags.

*angle*

[out] Reference to the variable for returning a font slope angle.

## FontNameGet

Receives font name.

```
string FontNameGet();
```

### Returned value

font name

## FontNameSet

Sets font name.

```
bool FontNameSet(  
    string name    // name  
);
```

### Parameters

*name*

[in] Font name. For example, "Arial".

### Returned value

true - if successful, otherwise - false

## FontSet

Sets the current font.

```
bool FontSet(  
    const string name,           // name  
    const int size,             // size  
    const uint flags=0,         // flags  
    const uint angle=0          // angle  
);
```

### Parameters

*name*

[in] Font name. For example, "Arial".

*size*

[in] Font size. See [TextSetFont\(\)](#) function description to learn more about setting a size.

*flags=0*

[in] Font creation flags. See [TextSetFont\(\)](#) function description to learn more about the flags.

*angle=0*

[in] Font slope angle in tenths of a degree.

### Returned value

true - if successful, otherwise - false

## FontSizeGet

Receives font size.

```
int FontSizeGet();
```

### Returned value

font size

## FontSizeSet

Sets font size.

```
bool FontSizeSet(  
    int size    // size  
);
```

### Parameters

*size*

[in] Font size. See [TextSetFont\(\)](#) function description to learn more about setting a size.

### Returned value

true - if successful, otherwise - false

## Height

Receives the height of a graphical resource.

```
int Height();
```

### Returned value

height of a graphical resource



## Line

Draws a segment of a freehand line.

```
void Line(  
    int      x1,      // X coordinate  
    int      y1,      // Y coordinate  
    int      x2,      // X coordinate  
    int      y2,      // Y coordinate  
    const uint clr     // color  
);
```

### Parameters

*x1*

[in] X coordinate of the segment's first point.

*y1*

[in] Y coordinate of the segment's first point.

*x2*

[in] X coordinate of the segment's second point.

*y2*

[in] Y coordinate of the segment's second point.

*clr*

[in] Color in ARGB format.

## LineAA

Draws a segment of a freehand line using antialiasing algorithm.

```
void LineAA(  
    const int   x1,           // X coordinate  
    const int   y1,           // Y coordinate  
    const int   x2,           // X coordinate  
    const int   y2,           // Y coordinate  
    const uint  clr,          // color  
    const uint  style=UINT_MAX // line style  
);
```

### Parameters

*x1*

[in] X coordinate of the segment's first point.

*y1*

[in] Y coordinate of the segment's first point.

*x2*

[in] X coordinate of the segment's second point.

*y2*

[in] Y coordinate of the segment's second point.

*clr*

[in] Color in ARGB format.

*style=UINT\_MAX*

[in] Line style is one of [ENUM\\_LINE\\_STYLE](#) enumeration's values or a custom value.

## LineHorizontal

Draws a segment of a horizontal line.

```
void LineHorizontal(  
    int      x1,      // X coordinate  
    int      x2,      // X coordinate  
    int      y,       // Y coordinate  
    const uint clr     // color  
);
```

### Parameters

*x1*

[in] Segment's X coordinate.

*x2*

[in] X coordinate of the segment's first point.

*y*

[in] Y coordinate of the segment's second point.

*clr*

[in] Color in ARGB format.

## LineStyleSet

Sets the line style.

```
void LineStyleSet(  
    const uint style    // style  
);
```

### Parameters

*style*

[in] Line style.

### Note

The input parameter can have any of ENUM\_LINE\_STYLE enumeration values. Besides, it is possible to create a custom line drawing style.

## LineVertical

Draws a segment of a vertical line.

```
void LineVertical(  
    int      x,          // X coordinate  
    int      y1,         // Y coordinate  
    int      y2,         // Y coordinate  
    const uint clr       // color  
) ;
```

### Parameters

*x*

[in] Segment's X coordinate.

*y1*

[in] Y coordinate of the segment's first point.

*y2*

[in] Y coordinate of the segment's second point.

*clr*

[in] Color in ARGB format.

## LoadFromFile

Reads an image from a BMP file.

```
bool LoadFromFile(  
    const string filename    // file name  
);
```

### Parameters

*filename*

[in] File name (including "BMP" extension).

### Returned value

true - if successful, otherwise - false

## PixelGet

Receives color of the point with the specified coordinates.

```
uint PixelGet(  
    const int x,      // X coordinate  
    const int y      // Y coordinate  
);
```

### Parameters

*x*

[in] Point's X coordinate.

*y*

[in] Point's Y coordinate.

### Returned value

Point color in ARGB format.

## PixelSet

Sets color of the point with the specified coordinates.

```
void PixelSet(  
    const int   x,           // X coordinate  
    const int   y,           // Y coordinate  
    const uint  clr          // color  
);
```

### Parameters

*x*

[in] Point's X coordinate.

*y*

[in] Point's Y coordinate.

*clr*

[in] Color in ARGB format.



## PixelSetAA

Draws a point using antialiasing algorithm.

```
void PixelSetAA(  
    const double x,      // X coordinate  
    const double y,      // Y coordinate  
    const uint   clr     // color  
);
```

### Parameters

*x*

[in] Point's X coordinate.

*y*

[in] Point's Y coordinate.

*clr*

[in] Color in ARGB format.

## Polygon

Draws a polygon.

```
void Polygon(  
    int&      x[],      // array of X coordinates  
    int&      y[],      // array of Y coordinates  
    const uint clr      // color  
);
```

### Parameters

*x[]*

[in] Array of X coordinates of a polygon points.

*y[]*

[in] Array of Y coordinates of a polygon points.

*clr*

[in] Color in ARGB format.

## PolygonAA

Draws a polygon using antialiasing algorithm.

```
void PolygonAA(  
    int&      x[],           // array of X coordinates  
    int&      y[],           // array of Y coordinates  
    const uint clr,          // color  
    const uint style=UINT_MAX // line style  
);
```

### Parameters

*x[]*

[in] Array of X coordinates of a polygon points.

*y[]*

[in] Array of Y coordinates of a polygon points.

*clr*

[in] Color in ARGB format.

*style=UINT\_MAX*

[in] Line style is one of [ENUM\\_LINE\\_STYLE](#) enumeration's values or a custom value.

## Polyline

Draws a polyline.

```
void Polyline(  
    int&      x[],      // array of X coordinates  
    int&      y[],      // array of Y coordinates  
    const uint clr      // color  
);
```

### Parameters

*x[]*

[in] Array of X coordinates of a polyline.

*y[]*

[in] Array of Y coordinates of a polyline.

*clr*

[in] Color in ARGB format.

## PolylineAA

Draws a polyline using antialiasing algorithm.

```
void PolylineAA(  
    int&      x[],           // array of X coordinates  
    int&      y[],           // array of Y coordinates  
    const uint clr,          // color  
    const uint style=UINT_MAX // line style  
);
```

### Parameters

*x[]*

[in] Array of X coordinates of a polyline.

*y[]*

[in] Array of Y coordinates of a polyline.

*clr*

[in] Color in ARGB format.

*style=UINT\_MAX*

[in] Line style is one of [ENUM\\_LINE\\_STYLE](#) enumeration's values or a custom value.

## Rectangle

Draws a rectangle using two points.

```
void Rectangle(  
    int      x1,      // X coordinate  
    int      y1,      // Y coordinate  
    int      x2,      // X coordinate  
    int      y2,      // Y coordinate  
    const uint clr     // color  
);
```

### Parameters

*x1*

[in] X coordinate of the first point forming a rectangle.

*y1*

[in] Y coordinate of the first point forming a rectangle.

*x2*

[in] X coordinate of the second point forming a rectangle.

*y2*

[in] Y coordinate of the second point forming a rectangle.

*clr*

[in] Color in ARGB format.

## Resize

Resizes a graphical resource.

```
bool  Resize(  
    const int  width,      // width  
    const int  height     // height  
);
```

### Parameters

*width*

[in] New width of a graphical resource.

*height*

[in] New height of a graphical resource.

### Returned value

true - if successful, otherwise - false

### Note

When resizing, the previous image is not saved.

## ResourceName

Receives the name of a graphical resource.

```
string ResourceName();
```

### Returned value

name of a graphical resource



## TextHeight

Receives the text height.

```
int TextHeight(  
    const string text    // text  
);
```

### Parameters

*text*

[in] Text for measuring.

### Returned value

text height in pixels

### Note

The current font is used for measuring the text.

## TextOut

Displays text.

```
void TextOut (
    int      x,           // X coordinate
    int      y,           // Y coordinate
    string    text,        // text
    const uint clr,        // color
    uint      alignment=0  // alignment
);
```

### Parameters

*x*

[in] Text anchor's X coordinate.

*y*

[in] Text anchor's Y coordinate.

*text*

[in] Text to be displayed.

*clr*

[in] Color in ARGB format.

*alignment=0*

[in] Text anchoring method. See [TextOut\(\)](#) function description to learn more about anchoring methods.

### Note

The current font is used to display the text.

## TextSize

Receives the text size.

```
void TextSize(  
    const string text,      // text  
    int& width,             // width  
    int& height            // height  
);
```

### Parameters

*text*

[in] Text for measuring.

*width*

[out] Reference to the variable for returning a text width.

*height*

[out] Reference to the variable for returning a text height.

### Note

The current font is used to measure the text.

## TextWidth

Receives the text width.

```
int TextWidth(  
    const string text    // text  
);
```

### Parameters

*text*

[in] Text for measuring.

### Returned value

text height in pixels

### Note

The current font is used to measure the text.

## TransparentLevelSet

Sets transparency level.

```
void TransparentLevelSet(  
    const uchar value    // value  
);
```

### Parameters

*value*

[in] New value of the transparency level.

### Note

0 stands for full transparency, while 255 - for full opacity.

Setting of a transparency level affects all that was previously drawn. The specified transparency level does not affect further constructions.

## Triangle

Draws a triangle.

```
void Triangle(  
    int      x1,      // X coordinate  
    int      y1,      // Y coordinate  
    int      x2,      // X coordinate  
    int      y2,      // Y coordinate  
    int      x3,      // X coordinate  
    int      y3,      // Y coordinate  
    const uint clr     // color  
);
```

### Parameters

*x1*

[in] X coordinate of the triangle's first corner.

*y1*

[in] Y coordinate of the triangle's first corner.

*x2*

[in] X coordinate of the triangle's second corner.

*y2*

[in] Y coordinate of the triangle's second corner.

*x3*

[in] X coordinate of the triangle's third corner.

*y3*

[in] Y coordinate of the triangle's third corner.

*clr*

[in] Color in ARGB format.

## TriangleAA

Draws a triangle using antialiasing algorithm.

```
void TriangleAA(  
    const int   x1,           // X coordinate  
    const int   y1,           // Y coordinate  
    const int   x2,           // X coordinate  
    const int   y2,           // Y coordinate  
    const int   x3,           // X coordinate  
    const int   y3,           // Y coordinate  
    const uint   clr,         // color  
    const uint   style=UINT_MAX // line style  
);
```

### Parameters

*x1*

[in] X coordinate of the triangle's first corner.

*y1*

[in] Y coordinate of the triangle's first corner.

*x2*

[in] X coordinate of the triangle's second corner.

*y2*

[in] Y coordinate of the triangle's second corner.

*x3*

[in] X coordinate of the triangle's third corner.

*y3*

[in] Y coordinate of the triangle's third corner.

*clr*

[in] Color in ARGB format.

*style=UINT\_MAX*

[in] Line style is one of [ENUM\\_LINE\\_STYLE](#) enumeration's values or a custom value.

## Update

Displays changes on the screen.

```
void Update(  
    const bool  redraw=true      // flag  
);
```

### Parameters

*redraw=true*

Flag of a chart redrawing necessity.



## Width

Receives the width of a graphical resource.

```
int Width();
```

### Returned value

graphical resource width

## CChart

Class CChart is a class for simplified access to "Chart" graphic object properties.

### Description

Class CChart provides access to "Chart" object properties.

### Declaration

```
class CChart : public CObject
```

### Title

```
#include <Charts\Chart.mqh>
```

### Class Methods

| Access to protected data     |                                                                              |
|------------------------------|------------------------------------------------------------------------------|
| <a href="#">ChartID</a>      | Gets identifier of the chart                                                 |
| General properties           |                                                                              |
| <a href="#">Mode</a>         | Gets/Sets the value of "Mode" property (bar, candle or line)                 |
| <a href="#">Foreground</a>   | Gets/Sets the value of "Foreground" property                                 |
| <a href="#">Shift</a>        | Gets/Sets the value of "Shift" property                                      |
| <a href="#">ShiftSize</a>    | Gets/Sets the value of "ShiftSize" property (in percents)                    |
| <a href="#">AutoScroll</a>   | Gets/Sets the value of "AutoScroll" property                                 |
| <a href="#">Scale</a>        | Gets/Sets the value of "Scale" property                                      |
| <a href="#">ScaleFix</a>     | Gets/Sets the value of "ScaleFix" property (fixed chart scale or not)        |
| <a href="#">ScaleFix_11</a>  | Gets/Sets the value of "ScaleFix_11" property (chart scale is 1:1, or not)   |
| <a href="#">FixedMax</a>     | Gets/Sets the value of "FixedMax" property (fixed maximal price)             |
| <a href="#">FixedMin</a>     | Gets/Sets the value of "FixedMin" property (fixed minimal price)             |
| <a href="#">ScalePPB</a>     | Gets/Sets the value of "ScalePPB" property (scale is "point per bar" or not) |
| <a href="#">PointsPerBar</a> | Gets/Sets the value of "PointsPerBar" property (in points per bar)           |
| Show properties              |                                                                              |

|                                        |                                                                                                           |
|----------------------------------------|-----------------------------------------------------------------------------------------------------------|
| <a href="#"><u>ShowOHLC</u></a>        | Gets/Sets the value of "ShowOHLC" property                                                                |
| <a href="#"><u>ShowLineBid</u></a>     | Gets/Sets the value of "ShowLineBid" property                                                             |
| <a href="#"><u>ShowLineAsk</u></a>     | Gets/Sets the value of "ShowLineAsk" property                                                             |
| <a href="#"><u>ShowLastLine</u></a>    | Gets/Sets the value of "ShowLastLine" property                                                            |
| <a href="#"><u>ShowPeriodSep</u></a>   | Gets/Sets the value of "ShowPeriodSep" property (show period separators)                                  |
| <a href="#"><u>ShowGrid</u></a>        | Gets/Sets the value of "ShowGrid" property                                                                |
| <a href="#"><u>ShowVolumes</u></a>     | Gets/Sets the value of "ColorVolumes" property (color for volumes and levels of opened positions)         |
| <a href="#"><u>ShowObjectDescr</u></a> | Gets/Sets the value of "ShowObjectDescr" property (show description for graphic objects)                  |
| <a href="#"><u>ShowDateScale</u></a>   | Sets the value of "ShowDateScale" property (date scale of the chart)                                      |
| <a href="#"><u>ShowPriceScale</u></a>  | Sets the value of "ShowPriceScale" property (price scale of the chart)                                    |
| <b>Colors properties</b>               |                                                                                                           |
| <a href="#"><u>ColorBackground</u></a> | Gets/Sets the value of "ColorBackground" property (background color of the chart)                         |
| <a href="#"><u>ColorForeground</u></a> | Gets/Sets the value of "ColorForeground" property (color of axes, scale and OHLC strings of the chart)    |
| <a href="#"><u>ColorGrid</u></a>       | Gets/Sets the value of "ColorGrid" property (color of the grid)                                           |
| <a href="#"><u>ColorBarUp</u></a>      | Gets/Sets the value of "ColorBarUp" property (color for bull bars, its shadow and candle body outlines)   |
| <a href="#"><u>ColorBarDown</u></a>    | Gets/Sets the value of "ColorBarDown" property (color for bear bars, its shadow and candle body outlines) |
| <a href="#"><u>ColorCandleBull</u></a> | Gets/Sets the value of "ColorCandleBull" property (body color of the bull candle)                         |
| <a href="#"><u>ColorCandleBear</u></a> | Gets/Sets the value of "ColorCandleBear" property (body color of the bear candle)                         |
| <a href="#"><u>ColorChartLine</u></a>  | Gets/Sets the value of "ColorChartLine" property (color for line chart and Doji candles)                  |
| <a href="#"><u>ColorVolumes</u></a>    | Gets/Sets the value of "ColorVolumes" property (color for volumes and levels of opened positions)         |

|                                          |                                                                                     |
|------------------------------------------|-------------------------------------------------------------------------------------|
| <a href="#"><u>ColorLineBid</u></a>      | Gets/Sets the value of "ColorLineBid" property (color of Bid line)                  |
| <a href="#"><u>ColorLineAsk</u></a>      | Gets/Sets the value of "ColorLineAsk" property (color of Ask line)                  |
| <a href="#"><u>ColorLineLast</u></a>     | Gets/Sets the value of "ColorLineLast" property (color of the last deal price line) |
| <a href="#"><u>ColorStopLevels</u></a>   | Gets/Sets the value of "ColorStopLevels" property (color of the SL and TP levels)   |
| <b>Read only properties</b>              |                                                                                     |
| <a href="#"><u>VisibleBars</u></a>       | Gets total number of visible chart bars                                             |
| <a href="#"><u>WindowsTotal</u></a>      | Gets total number of chart windows, including the chart indicator subwindows        |
| <a href="#"><u>WindowsVisible</u></a>    | Gets visibility flag of the specified chart subwindow                               |
| <a href="#"><u>WindowHandle</u></a>      | Gets window handle of the chart (HWND)                                              |
| <a href="#"><u>FirstVisibleBar</u></a>   | Gets the number of the first visible bar of the chart                               |
| <a href="#"><u>WidthInBars</u></a>       | Gets window width in bars.                                                          |
| <a href="#"><u>WidthInPixels</u></a>     | Gets subwindow width in pixels.                                                     |
| <a href="#"><u>HeightInPixels</u></a>    | Gets subwindow height in pixels.                                                    |
| <a href="#"><u>PriceMin</u></a>          | Gets minimal price of the specified subwindow                                       |
| <a href="#"><u>PriceMax</u></a>          | Gets maximal price of the specified subwindow                                       |
| <b>Properties</b>                        |                                                                                     |
| <a href="#"><u>Attach</u></a>            | Assigns the current chart to the class instance                                     |
| <a href="#"><u>FirstChart</u></a>        | Assigns the first chart of the client terminal to the class instance                |
| <a href="#"><u>NextChart</u></a>         | Assigns the next chart of the client terminal to the class instance                 |
| <a href="#"><u>Open</u></a>              | Opens chart with specified parameters and assign it to the class instance           |
| <a href="#"><u>Detach</u></a>            | Detaches chart from the class instance                                              |
| <a href="#"><u>Close</u></a>             | Closes chart, assigned to the class instance                                        |
| <a href="#"><u>BringToTop</u></a>        | Show chart on top of other charts                                                   |
| <a href="#"><u>EventObjectCreate</u></a> | Sets a flag to send notifications of an event of new object creation on a chart     |
| <a href="#"><u>EventObjectDelete</u></a> | Sets a flag to send notifications of an event of object deletion on a chart         |

|                                        |                                                                                       |
|----------------------------------------|---------------------------------------------------------------------------------------|
| <b>Indicators</b>                      |                                                                                       |
| <a href="#"><u>IndicatorAdd</u></a>    | Adds an indicator with the specified handle into a specified chart subwindow          |
| <a href="#"><u>IndicatorDelete</u></a> | Removes an indicator with a specified name from the specified chart subwindow         |
| <a href="#"><u>IndicatorsTotal</u></a> | Returns the number of all indicators applied to the specified chart subwindow         |
| <a href="#"><u>IndicatorName</u></a>   | Returns the short name of the indicator on the specified chart subwindow              |
| <b>Navigation</b>                      |                                                                                       |
| <a href="#"><u>Navigate</u></a>        | Navigates the chart                                                                   |
| <b>Access to MQL5 API</b>              |                                                                                       |
| <a href="#"><u>Symbol</u></a>          | Gets symbol of the chart                                                              |
| <a href="#"><u>Period</u></a>          | Gets period of the chart                                                              |
| <a href="#"><u>Redraw</u></a>          | Redraws chart, assigned to the class instance                                         |
| <a href="#"><u>GetInteger</u></a>      | The function returns the value of the corresponding object property                   |
| <a href="#"><u>SetInteger</u></a>      | Sets new value for the property of the integer type                                   |
| <a href="#"><u>GetDouble</u></a>       | The function returns the value of the corresponding object property                   |
| <a href="#"><u>SetDouble</u></a>       | Sets new value for the property of the double type                                    |
| <a href="#"><u>GetString</u></a>       | The function returns the value of the corresponding object property                   |
| <a href="#"><u>SetString</u></a>       | Sets new value for the property of the string type                                    |
| <a href="#"><u>SetSymbolPeriod</u></a> | Changes symbol and period of the chart, assigned to the class instance                |
| <a href="#"><u>ApplyTemplate</u></a>   | Applies specified template to the chart                                               |
| <a href="#"><u>ScreenShot</u></a>      | Creates screenshot of the specified chart and saves it to .gif file                   |
| <a href="#"><u>WindowOnDropped</u></a> | Gets chart subwindow number corresponding to the object (expert or script) drop point |
| <a href="#"><u>PriceOnDropped</u></a>  | Gets price coordinate corresponding to the object (expert or script) drop point       |
| <a href="#"><u>TimeOnDropped</u></a>   | Gets time coordinate corresponding to the object (expert or script) drop point        |

|                                     |                                                                             |
|-------------------------------------|-----------------------------------------------------------------------------|
| <a href="#"><u>XOnDropped</u></a>   | Gets X coordinate corresponding to the object (expert or script) drop point |
| <a href="#"><u>YOnDropped</u></a>   | Gets Y coordinate corresponding to the object (expert or script) drop point |
| <b>Input/Output</b>                 |                                                                             |
| virtual <a href="#"><u>Save</u></a> | Saves object parameters to file                                             |
| virtual <a href="#"><u>Load</u></a> | Loads object parameters from file                                           |
| virtual <a href="#"><u>Type</u></a> | Gets graphic object type identifier                                         |

## ChartID

Returns identifier of the chart.

```
long ChartID() const
```

### Returned value

Chart identifier, assigned to the class instance. If there is no object assigned, it returns -1.

## Mode (Get Method)

Gets the value of "Mode" property (bar, candle or line).

```
ENUM_CHART_MODE Mode() const
```

### Returned value

Value of "Mode" property of the object, assigned to the class instance. If there is no object assigned, it returns [WRONG\\_VALUE](#).

## Mode (Set Method)

Sets new value for "Mode" property (bar, candle or line).

```
bool Mode(  
    ENUM_CHART_MODE mode    // new chart mode  
)
```

### Parameters

*mode*

[in] Chart mode (candle, bar or line) of [ENUM\\_CHART\\_MODE](#) enumeration.

### Returned value

true if successful, false if mode hasn't changed.



## Foreground (Get Method)

Gets the value of "Foreground" property.

```
bool Foreground() const
```

### Returned value

Value of "Foreground" property of the object, assigned to the class instance. If there is no object assigned, it returns false.

## Foreground (Set Method)

Sets new value for "Foreground" property.

```
bool Foreground(  
    bool foreground    // new flag value  
)
```

### Parameters

*foreground*

[in] New value for "Foreground" property.

### Returned value

true if successful, false if property hasn't changed.

## Shift (Get Method)

Gets the value of "Shift" property.

```
bool Shift() const
```

### Returned value

Value of "Shift" property of the object, assigned to the class instance. If there is no object assigned, it returns false.

## Shift (Set Method)

Sets new value for "Shift" property.

```
bool Shift(  
    bool shift    // new flag value  
)
```

### Parameters

*shift*

[in] New value for "Shift" property.

### Returned value

true if successful, false if property hasn't changed.

## ShiftSize (Get Method)

Gets the value of "ShiftSize" property (in percents).

```
double ShiftSize() const
```

### Returned value

Value of "ShiftSize" property of the object, assigned to the class instance. If there is no object assigned, it returns [EMPTY\\_VALUE](#).

## ShiftSize (Set Method)

Sets new value for "Shift" property (in percents).

```
bool ShiftSize(  
    double shift_size    // new property value  
)
```

### Parameters

*shift\_size*

[in] New value for "ShiftSize" property (in percents).

### Returned value

true if successful, false if property hasn't changed.

## AutoScroll (Get Method)

Gets the value of "AutoScroll" property.

```
bool AutoScroll() const
```

### Returned value

Value of "AutoScroll" property of the object, assigned to the class instance. If there is no object assigned, it returns false.

## AutoScroll (Set Method)

Sets new value for "AutoScroll" property.

```
bool AutoScroll(  
    bool autoscroll    // new flag value  
)
```

### Parameters

*autoscroll*

[in] New value for "Autoscroll" property.

### Returned value

true if successful, false if property hasn't changed.

## Scale (Get Method)

Gets the value of "Scale" property.

```
int Scale() const
```

### Returned value

Value of "Scale" property of the object, assigned to the class instance. If there is no object assigned, it returns 0.

## Scale (Set Method)

Sets new value for "Scale" property.

```
bool Scale(  
    int scale    // new value  
)
```

### Parameters

*scale*

[in] New value for "Scale" property.

### Returned value

true if successful, false if property hasn't changed.

## ScaleFix (Get Method)

Gets the value of "ScaleFix" property (fixed chart scale or not).

```
bool ScaleFix() const
```

### Returned value

Value of "ScaleFix" property of the object, assigned to the class instance. If there is no object assigned, it returns false.

## ScaleFix (Set Method)

Sets new value for "ScaleFix" property.

```
bool ScaleFix(  
    bool scale_fix    // new value  
)
```

### Parameters

*scale\_fix*

[in] New value for "ScaleFix" property.

### Returned value

true if successful, false if property hasn't changed.

## ScaleFix\_11 (Get Method)

Gets the value of "ScaleFix\_11" property (chart scale is 1:1, or not).

```
bool ScaleFix_11() const
```

### Returned value

Value of "ScaleFix\_11" property of the object, assigned to the class instance. If there is no object assigned, it returns false.

## ScaleFix\_11 (Set Method)

Sets new value for "ScaleFix\_11" property.

```
bool ScaleFix_11(  
    string scale_11    // new value  
)
```

### Parameters

*scale\_11*

[in] New value for "ScaleFix\_11" property.

### Returned value

true if successful, false if property hasn't changed.

## FixedMax (Get Method)

Gets the value of "FixedMax" property (fixed maximal price).

```
double FixedMax() const
```

### Returned value

Value of "FixedMax" property of the object, assigned to the class instance. If there is no object assigned, it returns [EMPTY\\_VALUE](#).

## FixedMax (Set Method)

Sets new value for "FixedMax" property.

```
bool FixedMax(  
    double max    // new fixed maximum  
)
```

### Parameters

*max*

[in] New value for "FixedMax" property.

### Returned value

true if successful, false if property hasn't changed.



## FixedMin (Get Method)

Gets the value of "FixedMin" property (fixed minimal price).

```
double FixedMin() const
```

### Returned value

Value of "FixedMin" property of the object, assigned to the class instance. If there is no object assigned, it returns [EMPTY\\_VALUE](#).

## FixedMin (Set Method)

Sets new value for "FixedMin" property.

```
bool FixedMax (
    double min      // new fixed minimum
)
```

### Parameters

*max*

[in] New value for "FixedMin" property.

### Returned value

true if successful, false if property hasn't changed.

## PointsPerBar (Get Method)

Gets the value of "PointsPerBar" property (in points per bar).

```
double PointsPerBar() const
```

### Returned value

Value of "PointsPerBar" property of the object, assigned to the class instance. If there is no object assigned, it returns [EMPTY\\_VALUE](#).

## PointsPerBar (Set Method)

Sets new value for "PointsPerBar" property.

```
bool PointsPerBar (
    double ppb      // new scale (in points per bar)
)
```

### Parameters

*ppb*

[in] New value for scale (in points per bar).

### Returned value

true if successful, false if scale hasn't changed.

## ScalePPB (Get Method)

Gets the value of "ScalePPB" property (scale is "point per bar" or not).

```
bool ScalePPB() const
```

### Returned value

Value of "ScalePPB" property of the object, assigned to the class instance. If there is no object assigned, it returns false.

## ScalePPB (Set Method)

Sets new value for "ScalePPB" property.

```
bool ScalePPB(  
    bool scale_ppb    // new flag value  
)
```

### Parameters

*scale\_ppb*

[in] New value for "ScalePPB" property.

### Returned value

true if successful, false if property hasn't changed.

## ShowOHLC (Get Method)

Gets the value of "ShowOHLC" property.

```
bool ShowOHLC() const
```

### Returned value

Value of "ShowOHLC" property of the object, assigned to the class instance. If there is no object assigned, it returns false.

## ShowOHLC (Set Method)

Sets new value for "ShowOHLC" property.

```
bool ShowOHLC(  
    bool show    // new value  
)
```

### Parameters

*show*

[in] New value for "ShowOHLC" property.

### Returned value

true if successful, false if property hasn't changed.

## ShowLineBid (Get Method)

Gets the value of "ShowLineBid" property.

```
bool ShowLineBid() const
```

### Returned value

Value of "ShowLineBid" property of the chart, assigned to the class instance. If there is no chart assigned, it returns false.

## ShowLineBid (Set Method)

Sets new value for "ShowLineBid" property.

```
bool ShowLineBid(  
    bool show // new value  
)
```

### Parameters

*show*

[in] New value for "ShowLineBid" property.

### Returned value

true if successful, false if property hasn't changed.

## ShowLineAsk (Get Method)

Gets the value of "ShowLineAsk" property.

```
bool ShowLineAsk() const
```

### Returned value

Value of "ShowLineAsk" property of the chart, assigned to the class instance. If there is no chart assigned, it returns false.

## ShowLineAsk (Set Method)

Sets new value for "ShowLineAsk" property.

```
bool ShowLineAsk(  
    bool show // new value  
)
```

### Parameters

*show*

[in] New value for "ShowLineAsk" property.

### Returned value

true if successful, false if property hasn't changed.

## ShowLastLine (Get Method)

Gets the value of "ShowLastLine" property.

```
bool ShowLastLine() const
```

### Returned value

Value of "ShowLastLine" property of the chart, assigned to the class instance. If there is no chart assigned, it returns false.

## ShowLastLine (Set Method)

Sets new value for "ShowLastLine" property.

```
bool ShowLastLine (  
    bool show // new flag value  
)
```

### Parameters

*show*

[in] New value for "ShowLastLine" property.

### Returned value

true if successful, false if property hasn't changed.

## ShowPeriodSep (Get Method)

Gets the value of "ShowPeriodSep" property (show period separators).

```
bool ShowPeriodSep() const
```

### Returned value

Value of "ShowPeriodSep" property of the chart, assigned to the class instance. If there is no chart assigned, it returns false.

## ShowPeriodSep (Set Method)

Sets new value for "ShowPeriodSep" property.

```
bool ShowPeriodSep(  
    bool show // new value  
)
```

### Parameters

*show*

[in] New value for "ShowPeriodSep" property.

### Returned value

true if successful, false if property hasn't changed.



## ShowGrid (Get Method)

Gets the value of "ShowGrid" property.

```
bool ShowGrid() const
```

### Returned value

Value of "ShowGrid" property of the chart, assigned to the class instance. If there is no chart assigned, it returns false.

## ShowGrid (Set Method)

Sets new value for "ShowGrid" property.

```
bool ShowGrid(  
    bool show    // new value  
)
```

### Parameters

*show*

[in] New value for "ShowGrid" property.

### Returned value

true if successful, false if property hasn't changed.

## ShowVolumes (Get Method)

Gets the value of "ShowVolumes" property.

```
bool ShowVolumes() const
```

### Returned value

Value of "ShowVolumes" property of the chart, assigned to the class instance. If there is no chart assigned, it returns false.

## ShowVolumes (Set Method)

Sets new value for "ShowVolumes" property.

```
bool ShowVolumes (
    bool show    // new value
)
```

### Parameters

*show*

[in] New value for "ShowVolumes" property.

### Returned value

true if successful, false if property hasn't changed.

## ShowObjectDescr (Get Method)

Gets the value of "ShowObjectDescr" property (show description for graphic objects).

```
bool ShowObjectDescr() const
```

### Returned value

Value of "ShowObjectDescr" property of the chart, assigned to the class instance. If there is no chart assigned, it returns false.

## ShowObjectDescr (Set Method)

Sets new value for "ShowObjectDescr" property.

```
bool ShowObjectDescr(  
    bool show // New value  
)
```

### Parameters

*show*

[in] New value for "ShowObjectDescr" property.

### Returned value

true if successful, false if property hasn't changed.

## ShowDateScale

Sets new value for "ShowDateScale" property.

```
bool ShowDateScale(  
    bool show    // New value  
)
```

### Parameters

*show*

[in] New value for "ShowDateScale" property.

### Returned value

true if successful, false if property hasn't changed.

## ShowPriceScale

Sets new value for "ShowPriceScale" property.

```
bool ShowPriceScale(  
    bool show    // New value  
)
```

### Parameters

*show*

[in] New value for "ShowPriceScale" property.

### Returned value

true if successful, false if property hasn't changed.

## ColorBackground (Get Method)

Gets the value of "ColorBackground" property (background color of the chart).

```
color ColorBackground() const
```

### Returned value

Value of "ColorBackground" property of the chart, assigned to the class instance. If there is no chart assigned, it returns [CLR\\_NONE](#).

## ColorBackground (Set Method)

Sets new value for "ColorBackground" property.

```
bool ColorBackground(  
    color new_color    // new background color  
)
```

### Parameters

*new\_color*

[in] New background color.

### Returned value

true if successful, false if color hasn't changed.

## ColorForeground (Get Method)

Gets the value of "ColorForeground" property (color of axes, scale and OHLC strings of the chart).

```
color ColorForeground() const
```

### Returned value

Value of "ColorForeground" property of the chart, assigned to the class instance. If there is no chart assigned, it returns [CLR\\_NONE](#).

## ColorForeground (Set Method)

Sets new value for "ColorForeground" property (for axes, scale, and OHLC string).

```
bool ColorForeground(  
    color new_color    // New color  
)
```

### Parameters

*new\_color*

[in] New color for axes, scale and OHLC string.

### Returned value

true if successful, false if color hasn't changed.

## ColorGrid (Get Method)

Gets the value of "ColorGrid" property (color of the grid).

```
color ColorGrid() const
```

### Returned value

Value of "ColorGrid" property of the chart, assigned to the class instance. If there is no chart assigned, it returns [CLR\\_NONE](#).

## ColorGrid (Set Method)

Sets new value for "ColorGrid" property.

```
bool ColorGrid(  
    color new_color    // new grid color  
)
```

### Parameters

*new\_color*  
[in] New grid color.

### Returned value

true if successful, false if color hasn't changed.



## ColorBarUp (Get Method)

Gets the value of "ColorBarUp" property (color for bull bars, its shadow and candle body outlines).

```
color ColorBarUp() const
```

### Returned value

Value of "ColorBarUp" property of the chart, assigned to the class instance. If there is no chart assigned, it returns [CLR\\_NONE](#).

## ColorBarUp (Set Method)

Sets new value for "ColorBarUp" property.

```
bool ColorBarUp(  
    color new_color    // new color for bull bars  
)
```

### Parameters

*new\_color*

[in] New color for bull bars.

### Returned value

true if successful, false if color hasn't changed.

## ColorBarDown (Get Method)

Gets the value of "ColorBarDown" property (color for bear bars, its shadow and candle body outlines).

```
color ColorBarDown() const
```

### Returned value

Value of "ColorBarDown" property of the chart, assigned to the class instance. If there is no chart assigned, it returns [CLR\\_NONE](#).

## ColorBarDown (Set Method)

Sets new value for "ColorBarDown" property.

```
bool ColorBarDown (
    color new_color      // new color for bear bars
)
```

### Parameters

*new\_color*

[in] New color for bear bars.

### Returned value

true if successful, false if color hasn't changed.

## ColorCandleBull (Get Method)

Gets the value of "ColorCandleBull" property (body color of the bull candle).

```
color ColorCandleBull() const
```

### Returned value

Value of "ColorCandleBull" property of the chart, assigned to the class instance. If there is no chart assigned, it returns [CLR\\_NONE](#).

## ColorCandleBull (Set Method)

Sets new value for "ColorBarBull" property.

```
bool ColorCandleBull(  
    color new_color    // new color for bull candle body  
)
```

### Parameters

*new\_color*

[in] New color of the bull candle body.

### Returned value

true if successful, false if color hasn't changed.

## ColorCandleBear (Get Method)

Gets the value of "ColorCandleBear" property (body color of the bear candle).

```
color ColorCandleBear() const
```

### Returned value

Value of "ColorCandleBear" property of the chart, assigned to the class instance. If there is no chart assigned, it returns [CLR\\_NONE](#).

## ColorCandleBear (Set Method)

Sets new value for "ColorBarBear" property.

```
bool ColorCandleBear(  
    color new_color    // new color for bear candle body  
)
```

### Parameters

*new\_color*

[in] New color of the bear candle body.

### Returned value

true if successful, false if color hasn't changed.

## ColorChartLine (Get Method)

Gets the value of "ColorChartLine" property (color for line chart and Doji candles).

```
color ColorChartLine() const
```

### Returned value

Value of "ColorChartLine" property of the chart, assigned to the class instance. If there is no chart assigned, it returns [CLR\\_NONE](#).

## ColorChartLine (Set Method)

Sets new value for "ColorChartLine" property.

```
bool ColorChartLine(  
    color new_color    // new color of the chart lines  
)
```

### Parameters

*new\_color*

[in] New color of the chart lines (Doji candles).

### Returned value

true if successful, false if color hasn't changed.

## ColorVolumes (Get Method)

Gets the value of "ColorVolumes" property (color for volumes and levels of opened positions).

```
color ColorVolumes() const
```

### Returned value

Value of "ColorVolumes" property of the chart, assigned to the class instance. If there is no chart assigned, it returns [CLR\\_NONE](#).

## ColorVolumes (Set Method)

Sets new value for "ColorVolumes" property.

```
bool ColorVolumes (
    color new_color      // new color of the volumes (open position levels)
)
```

### Parameters

*new\_color*

[in] New color of the volumes (open position levels).

### Returned value

true if successful, false if color hasn't changed.

## ColorLineBid (Get Method)

Gets the value of "ColorLineBid" property (color of Bid line).

```
color ColorLineBid() const
```

### Returned value

Value of "ColorLineBid" property of the chart, assigned to the class instance. If there is no chart assigned, it returns [CLR\\_NONE](#).

## ColorLineBid (Set Method)

Sets new value for "ColorLineBid" property.

```
bool ColorLineBid(  
    color new_color    // new color for Bid line  
)
```

### Parameters

*new\_color*

[in] New color for Bid line.

### Returned value

true if successful, false if color hasn't changed.

## ColorLineAsk (Get Method)

Gets the value of "ColorLineAsk" property (color of Ask line).

```
color ColorLineAsk() const
```

### Returned value

Value of "ColorLineAsk" property of the chart, assigned to the class instance. If there is no chart assigned, it returns [CLR\\_NONE](#).

## ColorLineAsk (Set Method)

Sets new value for "ColorLineAsk" property.

```
bool ColorLineAsk(  
    color new_color    // new color for Ask line  
)
```

### Parameters

*new\_color*

[in] New color for Ask line.

### Returned value

true if successful, false if color hasn't changed.



## ColorLineLast (Get Method)

Gets the value of "ColorLineLast" property (color of the last deal price line).

```
color ColorLineLast() const
```

### Returned value

Value of "ColorLineLast" property of the chart, assigned to the class instance. If there is no chart assigned, it returns [CLR\\_NONE](#).

## ColorLineLast (Set Method)

Sets new value for "ColorLineLast" property.

```
bool ColorLineLast(  
    color new_color    // new color of the last deal price line  
)
```

### Parameters

*new\_color*

[in] New color of the last deal price line.

### Returned value

true if successful, false if color hasn't changed.

## ColorStopLevels (Get Method)

Gets the value of "ColorStopLevels" property (color of the SL and TP levels).

```
color ColorStopLevels() const
```

### Returned value

Value of "ColorStopLevels" property of the chart, assigned to the class instance. If there is no chart assigned, it returns [CLR\\_NONE](#).

## ColorStopLevels (Set Method)

Sets new value for "ColorStopLevels" property.

```
bool ColorStopLevels(  
    color new_color    // new color of the SL and TP price levels  
)
```

### Parameters

*new\_color*

[in] New color of the Stop Loss and Take Profit price levels.

### Returned value

true if successful, false if color hasn't changed.

## VisibleBars

Gets total number of visible chart bars.

```
int VisibleBars() const
```

### Returned value

Gets total number of visible bars of the chart, assigned to the class instance. If there is no chart assigned, it returns 0.

## WindowsTotal

Gets total number of chart windows, including the chart indicator subwindows.

```
int WindowsTotal() const
```

### Returned value

Total number of windows, including the chart indicator subwindows, assigned to the class instance.  
If there is no chart assigned, it returns 0.

## WindowIsVisible

Gets visibility flag of the specified chart subwindow.

```
bool WindowIsVisible(  
    int num      // subwindow number  
) const
```

### Parameters

*num*

[in] Subwindow number (0 means base window).

### Returned value

Returns visibility flag of the specified chart subwindow, assigned to the chart instance. If there is no chart assigned, it returns false.

## WindowHandle

Gets window handle of the chart (HWND).

```
int WindowHandle() const
```

### Returned value

Window handle of the chart, assigned to the chart instance. If there is no chart assigned, it returns [INVALID\\_HANDLE](#).

## FirstVisibleBar

Gets the number of the first visible bar of the chart.

```
int FirstVisibleBar() const
```

### Returned value

Number of the first visible bar of the chart, assigned to the chart instance. If there is no chart assigned, it returns -1.

## WidthInBars

Gets window width in bars.

```
int WidthInBars() const
```

### Returned value

Window width in chart bars, assigned to the chart instance. If there is no chart assigned, it returns 0.



## WidthInPixels

Gets subwindow width in pixels.

```
int WidthInPixels() const
```

### Returned value

Subwindow width in chart pixels, assigned to the chart instance. If there is no chart assigned, it returns 0.

## HeightInPixels

Gets subwindow height in pixels.

```
int HeightInPixels(  
    int num    // subwindow number  
) const
```

### Parameters

*num*

[in] Subwindow number (0 means base window).

### Returned value

Subwindow height in chart pixels, assigned to the chart instance. If there is no chart assigned, it returns 0.

## PriceMin

Gets minimal price of the specified subwindow.

```
double PriceMin(  
    int num    // subwindow number  
) const
```

### Parameters

*num*

[in] Subwindow number (0 means base window).

### Returned value

Minimal price value of the chart, assigned to the class instance. If there is not chart assigned, it returns [EMPTY\\_VALUE](#).

## PriceMax

Gets maximal price of the specified subwindow.

```
double PriceMax(  
    int num    // subwindow number  
) const
```

### Parameters

*num*

[in] Subwindow number (0 means base window).

### Returned value

Maximal price value of the chart, assigned to the class instance. If there is not chart assigned, it returns [EMPTY\\_VALUE](#).

## Attach

Assigns the current chart to the class instance.

```
void Attach()
```

## Attach

Assigns the specified chart to the class instance.

```
void Attach(  
    long chart    // Chart identifier  
)
```

### Parameters

*chart*

[in] Identifier of the chart to assign.

## FirstChart

Assigns the first chart of the client terminal to the class instance.

```
void FirstChart()
```

## NextChart

Assigns the next chart of the client terminal to the class instance.

```
void NextChart()
```

## Open

Opens chart with specified parameters and assign it to the class instance.

```
long Open(  
    const string      symbol_name,      // Symbol name  
    ENUM_TIMEFRAMES  timeframe         // Period  
)
```

### Parameters

*symbol\_name*

[in] Symbol name. [NULL](#) means the symbol of the current chart (to which expert attached).

*timeframe*

[in] Chart timeframe ([ENUM\\_TIMEFRAMES](#) enumeration). 0 means the current timeframe.

### Returned value

Chart identifier.



## Detach

Detaches chart from the class instance.

```
void Detach()
```

## Close

Closes chart, assigned to the class instance.

```
void Close()
```

## BringToTop

Show chart on top of other charts.

```
bool BringToTop() const
```

### Returned value

true if successful, false if error.

## EventObjectCreate

Sets a flag to send notifications of an [event](#) of new object creation to all MQL5-programs on a chart.

```
bool EventObjectCreate(  
    bool flag    // flag  
)
```

### Parameters

*flag*

[in] New flag value.

### Returned value

true - if successful, false - if flag hasn't been changed.

## EventObjectDelete

Sets a flag to send notifications of an [event](#) of object deletion to all MQL5-programs on a chart.

```
bool EventObjectDelete(  
    bool flag    // flag  
)
```

### Parameters

*flag*

[in] New flag value.

### Returned value

true - if successful, false - if flag hasn't been changed.

## IndicatorAdd

Adds an indicator with the specified handle into a specified chart window.

```
bool IndicatorAdd(  
    int    sub_win      // number of the sub-window  
    int    handle       // handle of the indicator  
);
```

### Parameters

*sub\_win*

[in] The number of the chart sub-window. 0 means the main chart window. if the number of a not-existing window is specified, a new window will be created.

*handle*

[in] The handle of the indicator.

### Return Value

The function returns true in case of success, otherwise it returns false. In order to obtain information about the [error](#), call the [GetLastError\(\)](#) function.

### See Also

[IndicatorDelete\(\)](#), [IndicatorsTotal\(\)](#), [IndicatorName\(\)](#).

## IndicatorDelete

Removes an indicator with a specified name from the specified chart window.

```
bool IndicatorDelete(  
    int          sub_win      // number of the subwindow  
    const string name         // short name of the indicator  
);
```

### Parameters

*sub\_win*

[in] Number of the chart subwindow. 0 denotes the main chart subwindow.

*const name*

[in] The short name of the indicator which is set in the [INDICATOR\\_SHORTNAME](#) property with the [IndicatorSetString\(\)](#) function. To get the short name of an indicator use the [IndicatorName\(\)](#) function.

### Return Value

Returns true in case of successful deletion of the indicator. Otherwise it returns false. To get [error](#) details use the [GetLastError\(\)](#) function.

### Note

If two indicators with identical short names exist in the chart subwindow, the first one in a row will be deleted.

If other indicators on this chart are based on the values of the indicator that is being deleted, such indicators will also be deleted.

Do not confuse the indicator short name and the file name that is specified when creating an indicator using functions [iCustom\(\)](#) and [IndicatorCreate\(\)](#). If the short name of an indicator is not set explicitly, then the name of the file containing the source code of the indicator will be specified during compilation.

Deletion of an indicator from a chart doesn't mean that its calculation part will be deleted from the terminal memory. To release the indicator handle use the [IndicatorRelease\(\)](#) function.

The indicator's short name should be formed correctly. It will be written to the [INDICATOR\\_SHORTNAME](#) property using the [IndicatorSetString\(\)](#) function. It is recommended that the short name should contain values of all the input parameters of the indicator, because the indicator to be deleted from the chart by the [IndicatorDelete\(\)](#) function is identified by the short name.

### See also

[IndicatorAdd\(\)](#), [IndicatorsTotal\(\)](#), [IndicatorName\(\)](#), [iCustom\(\)](#), [IndicatorCreate\(\)](#), [IndicatorSetString\(\)](#).

## IndicatorsTotal

Returns the number of all indicators applied to the specified chart window.

```
int IndicatorsTotal(  
    int sub_win    // number of the subwindow  
);
```

### Parameters

*sub\_win*

[in] Number of the chart subwindow. 0 denotes the main chart subwindow.

### Return Value

The number of indicators in the specified chart window. To get [error](#) details use the [GetLastError\(\)](#) function.

### Note

The function allows going searching through all the indicators attached to the chart. The number of all the windows of the chart can be obtained from the [CHART\\_WINDOWS\\_TOTAL](#) property using the [GetInteger\(\)](#) function.

### See also

[IndicatorAdd\(\)](#), [IndicatorDelete\(\)](#), [IndicatorsTotal\(\)](#), [iCustom\(\)](#), [IndicatorCreate\(\)](#), [IndicatorSetString\(\)](#).



## IndicatorName

Returns the short name of the indicator by the index in the indicators list on the specified chart window.

```
string IndicatorName(  
    int    sub_win    // number of the subwindow  
    int    index      // index of the indicator in the list of indicators added to the  
);
```

### Parameters

*sub\_win*

[in] Number of the chart subwindow. 0 denotes the main chart subwindow.

*index*

[in] the index of the indicator in the list of indicators. The numeration of indicators start with zero, i.e. the first indicator in the list has the 0 index. To obtain the number of indicators in the list use the [IndicatorsTotal\(\)](#) function.

### Return Value

The short name of the indicator which is set in the [INDICATOR\\_SHORTNAME](#) property with the [IndicatorSetString\(\)](#) function. To get [error](#) details use the [GetLastError\(\)](#) function.

### Note

Do not confuse the indicator short name and the file name that is specified when creating an indicator using functions [iCustom\(\)](#) and [IndicatorCreate\(\)](#). If the short name of an indicator is not set explicitly, then the name of the file containing the source code of the indicator will be specified during compilation.

The indicator's short name should be formed correctly. It will be written to the [INDICATOR\\_SHORTNAME](#) property using the [IndicatorSetString\(\)](#) function. It is recommended that the short name should contain values of all the input parameters of the indicator, because the indicator to be deleted from the chart by the [IndicatorDelete\(\)](#) function is identified by the short name.

### See also

[IndicatorAdd\(\)](#), [IndicatorDelete](#), [IndicatorsTotal](#), [iCustom\(\)](#), [IndicatorCreate\(\)](#), [IndicatorSetString\(\)](#).

## Navigate

Navigates the chart.

```
bool  Navigate(  
    ENUM_CHART_POSITION  position,      // Position  
    int                  shift=0       // Shift  
)
```

### Parameters

*position*

[in] Value of [ENUM\\_CHART\\_POSITION](#) enumeration.

*shift=0*

[in] Number of bars to shift.

### Returned value

true if successful, false if chart hasn't navigated.

## Symbol

Gets symbol of the chart.

```
string Symbol() const
```

### Returned value

Symbol of the chart, assigned to the class instance. If there is no chart assigned, it returns 0.

## Period

Gets period of the chart.

```
ENUM_TIMEFRAMES Period() const
```

### Returned value

Period of the chart, assigned to the class instance. If there is no chart assigned, it returns 0.

## Redraw

Redraws chart, assigned to the class instance.

```
void Redraw()
```

## GetInteger

The function returns the value of the corresponding object property. The object property must be of the integer type. There are 2 variants of the function.

### 1. Immediately returns the property value.

```
long GetInteger(  
    ENUM_CHART_PROPERTY_INTEGER prop_id,           // property identifier  
    int sub_window=0                               // subwindow number  
) const
```

### 2. If successful, puts the value of property to the specified variable of integer type, passed by reference as last parameter.

```
bool GetInteger(  
    ENUM_CHART_PROPERTY_INTEGER prop_id,           // property identifier  
    int sub_window,                               // subwindow number  
    long& value                                    // here we get the property value  
) const
```

### Parameters

*prop\_id*

[in] Property identifier ([ENUM\\_CHART\\_PROPERTY\\_INTEGER](#) enumeration).

*sub\_window*

[in] Chart subwindow number.

*value*

[in] Variable of the integer type that received the value of the requested property.

### Return Value

Value of property of the chart, assigned to the class instance. If there isn't any chart assigned, it returns -1.

For the second variant the function returns true, if this property is maintained and the value has been placed into the value variable, otherwise returns false. To read more about the [error](#) call [GetLastError\(\)](#).

## SetInteger

Sets new value for the property of the integer type.

```
bool SetInteger(  
    ENUM_CHART_PROPERTY_INTEGER prop_id,    // property identifier  
    long value                               // new value  
)
```

### Parameters

*prop\_id*

[in] Property identifier ([ENUM\\_CHART\\_PROPERTY\\_INTEGER](#) enumeration).

*value*

[in] New value of the property.

### Returned value

true if successful, false if property of the integer type hasn't changed.

## GetDouble

The function returns the value of the corresponding object property. The object property must be of the double type. There are 2 variants of the function.

### 1. Immediately returns the property value.

```
double  GetDouble (
    ENUM_CHART_PROPERTY_DOUBLE  prop_id,           // property identifier
    int                          sub_window=0      // subwindow number
) const
```

### 2. If successful, puts the value of property to the specified variable of double type, passed by reference as last parameter.

```
bool  GetDouble (
    ENUM_CHART_PROPERTY_DOUBLE  prop_id,           // property identifier
    int                          sub_window,       // subwindow number
    double&                     value             // here we get the property value
) const
```

### Parameters

*prop\_id*

[in] Property identifier ([ENUM\\_CHART\\_PROPERTY\\_DOUBLE](#) enumeration).

*sub\_window*

[in] Chart subwindow number.

*value*

[in] Variable of the double type that received the value of the requested property.

### Return Value

Value of property of the chart, assigned to the class instance. If there isn't any chart assigned, it returns EMPTY\_VALUE.

For the second variant the function returns true, if this property is maintained and the value has been placed into the value variable, otherwise returns false. To read more about the [error](#) call [GetLastError\(\)](#).



## SetDouble

Sets new value for the property of the double type.

```
bool SetDouble(  
    ENUM_CHART_PROPERTY_DOUBLE prop_id,    // property identifier  
    double value                            // new value  
)
```

### Parameters

*prop\_id*

[in] Property identifier ([ENUM\\_CHART\\_PROPERTY\\_DOUBLE](#) enumeration).

*value*

[in] New value for the property.

### Returned value

true if successful, false if property of the double type hasn't changed.

## GetString

The function returns the value of the corresponding object property. The object property must be of the string type. There are 2 variants of the function.

1. Immediately returns the property value.

```
string GetString(  
    ENUM_CHART_PROPERTY_STRING prop_id    // property identifier  
) const
```

2. If successful, puts the value of property to the specified variable of string type, passed by reference as last parameter.

```
bool GetString(  
    ENUM_CHART_PROPERTY_STRING prop_id,    // property identifier  
    string& value                          // here we get the property value  
) const
```

### Parameters

*prop\_id*

[in] Property identifier ([ENUM\\_CHART\\_PROPERTY\\_STRING](#) enumeration).

*sub\_window*

[in] Chart subwindow number.

*value*

[in] Variable of the string type that received the value of the requested property.

### Return Value

Value of property of the chart, assigned to the class instance. If there isn't any chart assigned, it returns "".

For the second variant the function returns true, if this property is maintained and the value has been placed into the value variable, otherwise returns false. To read more about the [error](#) call [GetLastError\(\)](#).

## SetString

Sets new value for the property of the string type.

```
bool SetString(  
    ENUM_CHART_PROPERTY_STRING prop_id,    // property identifier  
    string value                            // new property value  
)
```

### Parameters

*prop\_id*

[in] Property identifier ([ENUM\\_CHART\\_PROPERTY\\_STRING](#) enumeration).

*value*

[in] New value for the property.

### Returned value

true if successful, false if property of the string type hasn't changed.

## SetSymbolPeriod

Changes symbol and period of the chart, assigned to the class instance.

```
bool SetSymbolPeriod(  
    const string      symbol_name,      // Symbol  
    ENUM_TIMEFRAMES  timeframe         // Period  
)
```

### Parameters

*symbol\_name*

[in] New symbol name. NULL means the symbol of the current chart (to which expert attached).

*timeframe*

[in] New chart timeframe (ENUM\_TIMEFRAMES enumeration). 0 means the current timeframe.

### Returned value

true if successful, false if property hasn't changed.

## ApplyTemplate

Applies specified template to the chart.

```
bool ApplyTemplate(  
    const string filename    // template file name  
)
```

### Parameters

*filename*

[in] File name of the template.

### Returned value

true if successful, false if template hasn't applied.

## ScreenShot

Creates screenshot of the specified chart and saves it to .gif file.

```
bool  ScreenShot(  
    string      filename,           // File name  
    int         width,              // Width  
    int         height,             // Height  
    ENUM_ALIGN_MODE align_mode=ALIGN_RIGHT // Align type  
) const
```

### Parameters

*filename*

[in] File name for screenshot.

*width*

[in] Screenshot width in pixels.

*height*

[in] Screenshot height in pixels.

*align\_mode=ALIGN\_RIGHT*

[in] Align mode, if screenshot is narrow.

### Returned value

true if successful, false if error.

## WindowOnDropped

Gets chart subwindow number corresponding to the object (expert or script) drop point.

```
int WindowOnDropped() const
```

### Returned value

Chart subwindow number of the object drop point. 0 means main chart window.

## PriceOnDropped

Gets price coordinate corresponding to the object (expert or script) drop point.

```
double PriceOnDropped() const
```

### Returned value

Price coordinate of the object drop point.



## TimeOnDropped

Gets time coordinate corresponding to the object (expert or script) drop point.

```
datetime TimeOnDropped() const
```

### Returned value

Time coordinate of the object drop point.

## XOnDropped

Gets X coordinate corresponding to the object (expert or script) drop point.

```
int XOnDropped() const
```

### Returned value

X coordinate of the object drop point.

## YOnDropped

Gets Y coordinate corresponding to the object (expert or script) drop point.

```
int YOnDropped() const
```

### Returned value

Y coordinate of the object drop point.

## Save

Saves object parameters to file.

```
virtual bool Save(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] handle of the binary file already opened by [FileOpen\(...\)](#) function.

### Returned value

true if successful, false if error.

## Load

Loads object parameters from file.

```
virtual bool Load(  
    int file_handle    // File handle  
)
```

### Parameters

*file\_handle*

[in] handle of the binary file already opened by [FileOpen\(...\)](#) function.

### Returned value

true if successful, false if error.

## Type

Returns graphic object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier (0x1111 for CChart).

## File Operations

This section contains the technical details of the file operations classes and descriptions of the corresponding components of the standard MQL5 library.

The file operations classes use will save time in developing applications which uses file input/output operations.

The MQL5 Standard Library is placed in the working directory of the terminal in the Include\Files folder.

| Class                           | Description                  |
|---------------------------------|------------------------------|
| <a href="#"><u>CFile</u></a>    | Base file operations class   |
| <a href="#"><u>CFileBin</u></a> | Binary file operations class |
| <a href="#"><u>CFileTxt</u></a> | Text file operations class   |

## CFile

CFile is a base class for CFileBin and CFileTxt classes.

### Description

Class CFile provides the simplified access for all of its descendants to MQL5 API file and folder functions.

### Declaration

```
class CFile: public CObject
```

### Title

```
#include <Files\File.mqh>
```

### Class Methods

| Attributes                         |                                   |
|------------------------------------|-----------------------------------|
| <a href="#">Handle</a>             | Gets file handle                  |
| <a href="#">Filename</a>           | Gets file name                    |
| <a href="#">Flags</a>              | Gets file flags                   |
| <a href="#">SetUnicode</a>         | Sets/Clears the FILE_UNICODE flag |
| <a href="#">SetCommon</a>          | Sets/Clears the FILE_COMMON flag  |
| <b>General methods for files</b>   |                                   |
| <a href="#">Open</a>               | Opens file                        |
| <a href="#">Close</a>              | Closes file                       |
| <a href="#">Delete</a>             | Deletes file                      |
| <a href="#">IsExist</a>            | Checks file for existence         |
| <a href="#">Copy</a>               | Copies file                       |
| <a href="#">Move</a>               | Renames/moves file                |
| <a href="#">Size</a>               | Gets file size                    |
| <a href="#">Tell</a>               | Gets current file position        |
| <a href="#">Seek</a>               | Sets current file position        |
| <a href="#">Flush</a>              | Flushes data on disk              |
| <a href="#">IsEnding</a>           | Checks file for end               |
| <a href="#">IsLineEnding</a>       | Checks line for end               |
| <b>General methods for folders</b> |                                   |



|                               |                      |
|-------------------------------|----------------------|
| <a href="#">FolderCreate</a>  | Creates folder       |
| <a href="#">FolderDelete</a>  | Deletes folder       |
| <a href="#">FolderClean</a>   | Clears folder        |
| <b>Search methods</b>         |                      |
| <a href="#">FileFindFirst</a> | Begin file search    |
| <a href="#">FileFindNext</a>  | Continue file search |
| <a href="#">FileFindClose</a> | Close search handle  |

**Derived classes:**

- [CFileBin](#)
- [CFileTxt](#)

## Handle

Gets file handle of the opened file.

```
int Handle()
```

### Returned value

Handle of the opened file, assigned to the class instance. If there is no file assigned, it returns -1.

## FileName

Gets file name of the opened file.

```
string FileName()
```

### Returned value

File name of the opened file, assigned to the class instance. If there is no file assigned, it returns "".

## Flags

Gets flags of the opened file.

```
int  Flags ()
```

### Returned value

Flags of the opened file, assigned to the class instance.

## SetUnicode

Sets/Clears the FILE\_UNICODE flag.

```
void SetUnicode(  
    bool unicode    // New flag value  
)
```

### Parameters

*unicode*

[in] New value for FILE\_UNICODE flag.

### Note

The result of string operations is dependent on the FILE\_UNICODE flag. If it's false, the ANSI codes are used (one byte symbols). If it set, the UNICODE codes are used (two byte symbols). If the file has already opened, the flag cannot be changed.

## SetCommon

Sets/Clears the FILE\_COMMON flag.

```
void SetCommon(  
    bool common    // New flag value  
)
```

### Parameters

*common*

[in] New value for FILE\_COMMON flag.

### Note

The FILE\_UNICODE flag determines the current work folder. If it's false, the local terminal folder used as the current work folder. If it's true, the general folder used as the current work folder. If the file has already opened, the flag cannot be changed.

## Open

Open the specified file and if it successful, assigns it to the class instance.

```
int Open(  
    const string file_name,      // File name  
    int flags,                  // Flags  
    short delimiter=9           // Separator  
)
```

### Parameters

*file\_name*

[in] File name to open.

*flags*

[in] File open flags.

*delimiter=9*

[in] CSV file separator.

### Returned value

Handle of the opened file.

### Note

The work folder is dependent on the FILE\_COMMON flag, defined by SetCommon() method.

## Close

Closes file, assigned to the class instance.

```
void Close()
```



## Delete

Deletes the file, assigned to the file instance.

```
void Delete()
```

## Delete

Deletes the specified file.

```
void Delete(  
    const string file_name    // File name  
)
```

### Parameters

*file\_name*

[in] File name of the file to delete.

### Note

The work folder is dependent on FILE\_COMMON flag, defined by SetCommon() method.

## IsExist

Checks file for existence

```
bool IsExist(  
    const string file_name    // File name  
)
```

### Parameters

*file\_name*

[in] Name of the file to check.

### Returned value

true, if file exists.

## Copy

Copies a file.

```
bool Copy(  
    const string src_name,      // Source file name  
    int src_flag,              // Flag  
    const string dst_name,      // Destination file name  
    int dst_flags               // Flags  
)
```

### Parameters

*src\_name*

[in] File name of the file to copy.

*src\_flag*

[in] Flags of the file to copy (only FILE\_COMMON is used).

*dst\_name*

[in] File name of the destination file.

*dst\_flags*

[in] Flags of the destination file (only FILE\_REWRITE and FILE\_COMMON are used).

### Returned value

true if successful, false if it hasn't been copied.

## Move

Renames/moves file.

```
bool Move(  
    const string src_name,      // Source file name  
    int src_flag,              // Flag  
    const string dst_name,      // Destination file name  
    int dst_flags               // Flags  
)
```

### Parameters

*src\_name*

[in] File name of the file to move.

*src\_flag*

[in] Flags of the file to copy (only FILE\_COMMON is used).

*dst\_name*

[in] File name of the destination file.

*dst\_flags*

[in] Flags of the destination file (only FILE\_REWRITE and FILE\_COMMON are used).

### Returned value

true if successful, false if it hasn't been moved.

## Size

Gets file size in bytes.

```
ulong Size()
```

### Returned value

File size in bytes. If there isn't any file assigned, it returns ULONG\_MAX.

## Tell

Gets the current file position.

```
ulong Tell()
```

### Returned value

The current file position. If there isn't any file assigned, it returns ULONG\_MAX.

## Seek

Sets current file position.

```
void Seek(  
    long          offset,      // Offset  
    ENUM_FILE_POSITION origin  // Origin  
)
```

### Parameters

*offset*

[in] File offset in bytes (can be negative).

*origin*

[in] Origin of the offset.

### Returned value

true if successful, false if file position hasn't been changed.

## Flush

Flushes all of the file input/output buffer data on disk.

```
void Flush()
```



## IsEnding

Checks file for end. It's used during the file read operations.

```
bool IsEnding()
```

### Returned value

true if end of file has been achieved after read or seek operation.

## IsLineEnding

Checks file for end of line. It's used during the file read operations.

```
bool IsLineEnding()
```

### Returned value

true if end of line has been achieved after the txt or csv file read operation (CR-LF chars).

## FolderCreate

Creates new folder.

```
bool FolderCreate(  
    const string folder_name    // Folder name  
)
```

### Parameters

*folder\_name*

[in] Name of the folder to create. It contains path to the folder relative to the folder defined by FILE\_COMMON flag.

### Returned value

true if successful, and false if the folder hasn't been created.

### Note

The work folder is dependent on FILE\_COMMON flag, defined by SetCommon() method.

## FolderDelete

Deletes specified folder.

```
bool FolderDelete(  
    const string folder_name    // Folder name  
)
```

### Parameters

*folder\_name*

[in] Name of the folder to delete. It contains path to the folder relative to the folder defined by FILE\_COMMON flag.

### Returned value

true if successful, and false if the folder hasn't been deleted.

### Note

The work folder is dependent on FILE\_COMMON flag, defined by SetCommon() method.

## FolderClean

Cleans specified folder.

```
bool FolderClean(  
    const string folder_name    // Folder name  
)
```

### Parameters

*folder\_name*

[in] Name of the folder to delete. It contains path to the folder relative to the folder defined by FILE\_COMMON flag.

### Returned value

true if successful, and false if the folder hasn't been cleaned.

### Note

The work folder is dependent on FILE\_COMMON flag, defined by SetCommon() method.

## FileFindFirst

It began file search using the filter specified.

```
int FileFindFirst(  
    const string filter,          // Search Filter  
    string&      file_name       // Reference to string  
)
```

### Parameters

*filter*

[in] Search filter.

*file\_name*

[out] The reference to string for the first file found.

### Returned value

If successful, it returns handle, that can be used for further file search using FileFindNext, or INVALID\_HANDLE if there isn't any file corresponding to the filter specified.

### Note

The work folder is dependent on FILE\_COMMON flag, defined by SetCommon() method.

## FileFindNext

It continue search, started by function FileFindFirst().

```
bool FileFindNext (
    int      search_handle,      // Search handle
    string&   file_name          // Reference to string for the next file found
)
```

### Parameters

*search\_handle*

[in] Search handle, returned by FileFindFirst() method.

*file\_name*

[in] The reference to string for the name of the file found if successful.

### Returned value

true if successful, false if there isn't any file, corresponding to the filter specified.

## FileFindClose

Closes search handle.

```
void FileFindClose(  
    int search_handle    // Search handle  
)
```

### Parameters

*search\_handle*

[in] Search handle, returned by FileFindFirst() method.



## CFileBin

CFileBin is a class for simplified access to binary files.

### Description

Class CFileBin provides an access to binary files.

### Declaration

```
class CFileBin: public CFile
```

### Title

```
#include <Files\FileBin.mqh>
```

### Class Methods

| Open methods                      |                                                     |
|-----------------------------------|-----------------------------------------------------|
| <a href="#">Open</a>              | Opens a binary file                                 |
| Write methods                     |                                                     |
| <a href="#">WriteChar</a>         | Writes char or uchar type variable                  |
| <a href="#">WriteShort</a>        | Writes short or ushort type variable                |
| <a href="#">WriteInteger</a>      | Writes int or uint type variable                    |
| <a href="#">WriteLong</a>         | Writes long or ulong type variable                  |
| <a href="#">WriteFloat</a>        | Writes float type variable                          |
| <a href="#">WriteDouble</a>       | Writes double type variable                         |
| <a href="#">WriteString</a>       | Writes string type variable                         |
| <a href="#">WriteCharArray</a>    | Writes an array of char or uchar type variables     |
| <a href="#">WriteShortArray</a>   | Writes an array of short or ushort type variables   |
| <a href="#">WriteIntegerArray</a> | Writes an array of int or uint type variables       |
| <a href="#">WriteLongArray</a>    | Writes an array of long or ulong type variables     |
| <a href="#">WriteFloatArray</a>   | Writes an array of float variables                  |
| <a href="#">WriteDoubleArray</a>  | Writes an array of double type variables            |
| <a href="#">WriteObject</a>       | Writes data of the CObject class inheritor instance |
| Read methods                      |                                                     |
| <a href="#">ReadChar</a>          | Reads char or uchar type variable                   |
| <a href="#">ReadShort</a>         | Reads short or ushort type variable                 |

|                                         |                                                    |
|-----------------------------------------|----------------------------------------------------|
| <a href="#"><u>ReadInteger</u></a>      | Reads int or uint type variable                    |
| <a href="#"><u>ReadLong</u></a>         | Reads long or ulong type variable                  |
| <a href="#"><u>ReadFloat</u></a>        | Reads float type variable                          |
| <a href="#"><u>ReadDouble</u></a>       | Reads double type variable                         |
| <a href="#"><u>ReadString</u></a>       | Reads string type variable                         |
| <a href="#"><u>ReadCharArray</u></a>    | Reads an array of char or uchar type variables     |
| <a href="#"><u>ReadShortArray</u></a>   | Reads an array of short or ushort type variables   |
| <a href="#"><u>ReadIntegerArray</u></a> | Reads an array of int or uint type variables       |
| <a href="#"><u>ReadLongArray</u></a>    | Reads an array of long or ulong type variables     |
| <a href="#"><u>ReadFloatArray</u></a>   | Reads an array of float type variables             |
| <a href="#"><u>ReadDoubleArray</u></a>  | Reads an array of double type variables            |
| <a href="#"><u>ReadObject</u></a>       | Reads data of the CObject class inheritor instance |

## Open

Open the specified binary file and if it successful, assigns it to the class instance.

```
int Open(  
    const string file_name,    // File name  
    int flags                 // Flags  
)
```

### Parameters

*file\_name*

[in] File name of the file to open.

*flags*

[in] File open flags (there is a forced set of the FILE\_BIN flag).

### Returned value

Handle of the opened file.

## WriteChar

Writes char or uchar type variable to file.

```
uint WriteChar(  
    char value    // Value  
)
```

### Parameters

*value*

[in] Variable to write.

### Returned value

Number of bytes written.

## WriteShort

Writes short or ushort type variable to file.

```
uint WriteShort(  
    short value    // Value  
)
```

### Parameters

*value*

[in] Variable to write.

### Returned value

Number of bytes written.

## WriteInteger

Writes int or uint type variable to file.

```
uint WriteInteger(  
    int value    // Value  
)
```

### Parameters

*value*

[in] Variable to write.

### Returned value

Number of bytes written.

## WriteLong

Writes long or ulong type variable to file.

```
uint WriteLong(  
    long value    // Value  
)
```

### Parameters

*value*

[in] Variable to write.

### Returned value

Number of bytes written.

## WriteFloat

Writes float type variable to file.

```
uint WriteFloat(  
    float value    // Value  
)
```

### Parameters

*value*

[in] Variable to write.

### Returned value

Number of bytes written.



## WriteDouble

Writes double type variable to file.

```
uint WriteDouble(  
    double value    // Value  
)
```

### Parameters

*value*

[in] Variable to write.

### Returned value

Number of bytes written.

## WriteString

Writes string type variable to file.

```
uint WriteString(  
    const string value    // Value  
)
```

### Parameters

*value*

[in] String to write.

### Returned value

Number of bytes written.

## WriteString

Writes string type variable to file.

```
uint WriteString(  
    const string value,    // Value  
    int size              // Size  
)
```

### Parameters

*value*

[in] String to write.

*size*

[in] Number of bytes to write.

### Returned value

Number of bytes written.

## WriteCharArray

Writes an array of char or uchar type variables to file.

```
uint WriteCharArray(  
    char& array[],           // Array reference  
    int start_item=0,        // Start element  
    int items_count=-1       // Number of elements  
)
```

### Parameters

*array[]*

[in] Array to write.

*start\_item=0*

[in] Start element to write from.

*items\_count=-1*

[in] Number of elements to write (-1 - for whole array).

### Returned value

Number of bytes written.

## WriteShortArray

Writes an array of short or ushort type variables to file.

```
uint WriteShortArray(  
    short& array[],           // Array to write  
    int start_item=0,         // Start element  
    int items_count=-1        // Number of elements to write  
)
```

### Parameters

*array[]*

[in] Array to write.

*start\_item=0*

[in] Start element to write from.

*items\_count=-1*

[in] Number of elements to write (-1 - for whole array).

### Returned value

Number of bytes written.

## WriteIntegerArray

Writes an array of int or uint type variables to file.

```
uint WriteIntegerArray(  
    int& array[],           // Array to write  
    int start_item=0,       // Start element  
    int items_count=-1      // Number of elements to write  
)
```

### Parameters

*array[]*

[in] Array to write.

*start\_item=0*

[in] Start element to write from.

*items\_count=-1*

[in] Number of elements to write (-1 - for whole array).

### Returned value

Number of bytes written.

## WriteLongArray

Writes an array of long or ulong type variables to file.

```
uint WriteLongArray(  
    long& array[],           // Array to write  
    int start_item=0,       // Start element  
    int items_count=-1      // Number of elements to write  
)
```

### Parameters

*array[]*

[in] Array to write.

*start\_item=0*

[in] Start element to write from.

*items\_count=-1*

[in] Number of elements to write (-1 - for whole array).

### Returned value

Number of bytes written.

## WriteFloatArray

Writes an array of float type variables to file.

```
uint WriteFloatArray(  
    float& array[],           // Array to write  
    int    start_item=0,      // Start element  
    int    items_count=-1     // Number of elements to write  
)
```

### Parameters

*array[]*

[in] Array to write.

*start\_item=0*

[in] Start element to write from.

*items\_count=-1*

[in] Number of elements to write (-1 - for whole array).

### Returned value

Number of bytes written.

## WriteDoubleArray

Writes an array of double type variables to file.

```
uint WriteDoubleArray(  
    double& array[],           // Array to write  
    int start_item=0,          // Start element  
    int items_count=-1         // Number of elements to write  
)
```

### Parameters

*array[]*

[in] Array to write.

*start\_item=0*

[in] Start element to write from.

*items\_count=-1*

[in] Number of elements to write (-1 - for whole array).

### Returned value

Number of bytes written.



## WriteObject

Writes data of the CObject class inheritor instance to file.

```
bool WriteObject(  
    CObject* object    // Reference to the object  
)
```

### Parameters

*object*

[in] Reference to the CObject class inheritor instance to write.

### Returned value

true if successful, false if data hasn't been written.

## ReadChar

Reads char or uchar type variable from file.

```
bool ReadChar(  
    char& value    // Target variable  
)
```

### Parameters

*value*

[in] Target variable of type char.

### Returned value

true if successful, false if data hasn't been read.

## ReadShort

Reads short or ushort type variable from file.

```
bool ReadShort(  
    short& value  
)
```

### Parameters

*value*

[in] Target variable of type short or ushort.

### Returned value

true if successful, false if data hasn't been read.

## ReadInteger

Reads int or uint type variable from file.

```
bool ReadInteger(  
    int& value    // Target variable  
)
```

### Parameters

*value*

[in] Target variable of type int or uint.

### Returned value

true if successful, false if data hasn't been read.

## ReadLong

Reads long or ulong type variable from file.

```
bool ReadLong(  
    long& value  
)
```

### Parameters

*value*

[in] Target variable of type long or ulong.

### Returned value

true if successful, false if data hasn't been read.

## ReadFloat

Reads float type variable from file.

```
bool ReadFloat(  
    float& value    // Target variable  
)
```

### Parameters

*value*

[in] Target variable of type float.

### Returned value

true if successful, false if data hasn't been read.

## ReadDouble

Reads double type variable from file.

```
bool ReadDouble(  
    double& value  
)
```

### Parameters

*value*

[in] Target variable of type double.

### Returned value

true if successful, false if data hasn't been read.

## ReadString

Reads string type variable from file.

```
bool ReadString(  
    string& value      // Target string  
)
```

### Parameters

*value*

[in] Target variable of type string.

### Returned value

true if successful, false if data hasn't been read.

## ReadString

Reads string type variable from file.

```
bool ReadString(  
    string& value  
)
```

### Parameters

*value*

[in] Target variable of type string.

### Returned value

true if successful, false if data hasn't been read.



## ReadCharArray

Reads an array of char or uchar type variables from file.

```
bool ReadCharArray(  
    char& array[],           // Target array  
    int start_item=0,        // Start element  
    int items_count=-1       // Number of elements to read  
)
```

### Parameters

*array[]*

[in] Reference to the target array of type char or uchar.

*start\_item=0*

[in] Start element to read from.

*items\_count=-1*

[in] Number of elements to read (-1 - read to the end of file).

### Returned value

true if successful, false if data hasn't been read.

## ReadShortArray

Reads an array of short or ushort type variables from file.

```
bool ReadShortArray(  
    short& array[],           // Target array  
    int start_item=0,        // Start element  
    int items_count=-1       // Number of elements to read  
)
```

### Parameters

*array[]*

[in] Reference to the target array of type short or ushort.

*start\_item=0*

[in] Start element to read from.

*items\_count=-1*

[in] Number of elements to read (-1 - read to the end of file).

### Returned value

true if successful, false if data hasn't been read.

## ReadIntegerArray

Reads an array of int or uint type variables from file.

```
bool ReadIntegerArray(  
    int& array[],           // Target array  
    int start_item=0,       // Start element  
    int items_count=-1      // Number of elements to read  
)
```

### Parameters

*array[]*

[in] Reference to the target array of type int or uint.

*start\_item=0*

[in] Start element to read from.

*items\_count=-1*

[in] Number of elements to read (-1 - read to the end of file).

### Returned value

true if successful, false if data hasn't been read.

## ReadLongArray

Reads an array of long or ulong type variables from file.

```
bool ReadLongArray(  
    long& array[],           // Target array  
    int start_item=0,       // Start element  
    int items_count=-1      // Number of elements to read  
)
```

### Parameters

*array[]*

[in] Reference to the target array of type long or ulong.

*start\_item=0*

[in] Start element to read from.

*items\_count=-1*

[in] Number of elements to read (-1 - read to the end of file).

### Returned value

true if successful, false if data hasn't been read.

## ReadFloatArray

Reads an array of float type variables from file.

```
bool ReadFloatArray(  
    float& array[],           // Target array  
    int start_item=0,         // Start element  
    int items_count=-1       // Number of elements to read  
)
```

### Parameters

*array[]*

[in] Reference to the target array of type float.

*start\_item=0*

[in] Start element to read from.

*items\_count=-1*

[in] Number of elements to read (-1 - read to the end of file).

### Returned value

true if successful, false if data hasn't been read.

## ReadDoubleArray

Reads an array of double type variables from file.

```
bool ReadDoubleArray(  
    double& array[],           // Target array  
    int start_item=0,         // Start element  
    int items_count=-1        // Number of elements to read  
)
```

### Parameters

*array[]*

[in] Reference to the target array of type double.

*start\_item=0*

[in] Start element to read from.

*items\_count=-1*

[in] Number of elements to read (-1 - read to the end of file).

### Returned value

true if successful, false if data hasn't been read.

## ReadObject

Reads data of the CObject class inheritor instance from file.

```
bool ReadObject(  
    CObject* object    // Reference to the object  
)
```

### Parameters

*object*

[in] Reference to the target CObject class inheritor instance for read to.

### Returned value

true if successful, false if data hasn't been read.

## CFileTxt

CFileTxt is a class for simplified access to text files.

### Description

Class CFileTxt provides an access to text files.

### Declaration

```
class CFileTxt: public CFile
```

### Title

```
#include <Files\FileTxt.mqh>
```

### Class Methods

| Open methods                |                                      |
|-----------------------------|--------------------------------------|
| <a href="#">Open</a>        | Open a text file                     |
| Write methods               |                                      |
| <a href="#">WriteString</a> | Writes string type variable to file  |
| Read methods                |                                      |
| <a href="#">ReadString</a>  | Reads string type variable from file |



## Open

Open the specified text file and if it successful, assigns it to the class instance.

```
int Open(  
    const string file_name,    // file name  
    int flags                 // flags  
)
```

### Parameters

*file\_name*

[in] File name to open.

*flags*

[in] File open flags (there is a forced set of the FILE\_TXT flag).

### Returned value

Opened file handle.

## WriteString

Writes string type variable to file.

```
uint WriteString(  
    const string value    // String to write  
)
```

### Parameters

*value*

[in] String to write.

### Returned value

Number of bytes written.

## ReadString

Reads string type variable from file.

```
string ReadString()
```

### Returned value

String which has been read.

## String operations

This section contains the technical details of the string operations classes and descriptions of the corresponding components of the standard MQL5 library.

The use of string operations classes will save time in developing applications which uses text processing operations.

The MQL5 Standard Library is placed in the working directory of the terminal in the Include\Strings folder.

| Class                          | Description                 |
|--------------------------------|-----------------------------|
| <a href="#"><u>CString</u></a> | Class for string operations |

## CString

CString is a class for simplified access to the variables of string type.

### Description

Class CFile provides the simplified access for all of its descendants to MQL5 API string functions.

### Declaration

```
class CString: public CObject
```

### Title

```
#include <Strings\String.mqh>
```

### Class Methods

| Data access methods           |                                                                                             |
|-------------------------------|---------------------------------------------------------------------------------------------|
| <a href="#">Str</a>           | Gets a string                                                                               |
| <a href="#">Len</a>           | Gets length of a string                                                                     |
| <a href="#">Copy</a>          | Copies a string                                                                             |
| Fill methods                  |                                                                                             |
| <a href="#">Fill</a>          | Fills a string with specified char                                                          |
| <a href="#">Assign</a>        | Assigns a string                                                                            |
| <a href="#">Append</a>        | Appends a string                                                                            |
| <a href="#">Insert</a>        | Inserts a string                                                                            |
| Compare methods               |                                                                                             |
| <a href="#">Compare</a>       | Compares a string                                                                           |
| <a href="#">CompareNoCase</a> | Compares a strings case insensitive                                                         |
| Substring methods             |                                                                                             |
| <a href="#">Left</a>          | Gets a specified number of characters from the left side of a string                        |
| <a href="#">Right</a>         | Gets a specified number of characters from the right side of a string                       |
| <a href="#">Mid</a>           | Gets a specified number of characters from a string                                         |
| Trim/delete methods           |                                                                                             |
| <a href="#">Trim</a>          | Removes all leading and trailing occurrences of a set of specified characters from a string |

|                           |                                                                                 |
|---------------------------|---------------------------------------------------------------------------------|
| <a href="#">TrimLeft</a>  | Removes all leading occurrences of a set of specified characters from a string  |
| <a href="#">TrimRight</a> | Removes all trailing occurrences of a set of specified characters from a string |
| <a href="#">Clear</a>     | Clears a string                                                                 |
| <b>Convert methods</b>    |                                                                                 |
| <a href="#">ToUpper</a>   | Converts a string to uppercase.                                                 |
| <a href="#">ToLower</a>   | Converts a string to lowercase.                                                 |
| <a href="#">Reverse</a>   | Reverses a string                                                               |
| <b>Search methods</b>     |                                                                                 |
| <a href="#">Find</a>      | Searches for the first match of a substring                                     |
| <a href="#">FindRev</a>   | Searches for the last match of a substring                                      |
| <a href="#">Remove</a>    | Deletes a substring from a string                                               |
| <a href="#">Replace</a>   | Replaces a substring                                                            |

## Str

Gets a string.

```
string Str() const;
```

### Returned value

Copy of a string.

## Len

Gets length of a string.

```
uint Len() const;
```

### Returned value

Length of a string.



## Copy

Copies a string by reference.

```
void Copy(  
    string& copy      // Reference  
    ) const;
```

### Parameters

*copy*

[in] Reference to a string to copy.

## Copy

Copies a string to the CString class instance.

```
void Copy(  
    CString* copy      // Object descriptor  
    ) const;
```

### Parameters

*copy*

[in] CString class object descriptor.

## Fill

Fills a string with specified char.

```
bool Fill(  
    short character    // Character  
)
```

### Parameters

*character*

[in] Character for filling.

### Returned value

true if successful, false if a string hasn't been filled.

## Assign

Assigns a string.

```
void Assign(  
    const string str    // String to assign  
)
```

### Parameters

*str*

[in] String to assign.

## Assign

Assigns a string to the CString class instance.

```
void Assign(  
    CString* str        // Object descriptor  
)
```

### Parameters

*str*

[in] CString class object descriptor to assign.

## Append

Appends a string.

```
void Append(  
    const string str      // String to append  
)
```

### Parameters

*str*

[in] String to append.

## Append

Appends a string to the CString class instance.

```
void Append(  
    CString* string      // Object descriptor  
)
```

### Parameters

*string*

[in] CString class object descriptor to append.

## Insert

Inserts a string to the specified position.

```
uint Insert(  
    uint      pos,      // Position  
    const string str    // String to insert  
)
```

### Parameters

*pos*

[in] Insert position.

*str*

[in] String to insert.

### Returned value

Resulted string length.

## Insert

Inserts a string to the specified position to the CString class instance.

```
uint Insert(  
    uint      pos,      // Position  
    CString*  str       // Object descriptor  
)
```

### Parameters

*pos*

[in] Insert position.

*str*

[in] CString class object descriptor to insert.

### Returned value

Resulted string length.

## Compare

Compares a string.

```
int Compare(  
    const string str    // String to compare  
    ) const;
```

### Parameters

*str*

[in] String to compare.

### Returned value

I returns 0 if a strings are equal, -1 if a string of the class is lower than a string to compare, -1 if the class string greater than a string to compare.

## Compare

Compares a string with a string of the CString class instance.

```
int Compare(  
    CString* str        // Object descriptor  
    ) const;
```

### Parameters

*str*

[in] CString class object descriptor to compare.

### Returned value

I returns 0 if a strings are equal, -1 if a string of the class is lower than a string to compare, -1 if the class string greater than a string to compare.

## CompareNoCase

Compares a strings case insensitive.

```
int CompareNoCase(  
    const string str      // String to compare  
    ) const;
```

### Parameters

*str*

[in] String to compare.

### Returned value

It returns 0 if a strings are equal, -1 if a string of the class is lower than a string to compare, -1 if the class string greater than a string to compare.

## CompareNoCase

Compares a string (case insensitive) with a string of the CString class instance.

```
int CompareNoCase(  
    CString* str          // Object descriptor  
    ) const;
```

### Parameters

*str*

[in] CString class object descriptor to compare.

### Returned value

It returns 0 if a strings are equal, -1 if a string of the class is lower than a string to compare, -1 if the class string greater than a string to compare.

## Left

Gets a specified number of characters from the left side of a string.

```
string Left(  
    uint count    // Number of characters  
)
```

### Parameters

*count*

[in] Number of characters.

### Returned value

Resulted substring.



## Right

Gets a specified number of characters from the right side of a string.

```
string Right(  
    uint count    // Number of characters  
)
```

### Parameters

*count*

[in] Number of characters.

### Returned value

Resulted substring.

## Mid

Gets a specified number of characters from a string.

```
string Mid(  
    uint pos,           // Position  
    uint count          // Number of characters  
)
```

### Parameters

*pos*

[in] Position of a string.

*count*

[in] Number of characters.

### Returned value

Resulted substring.

## Trim

Removes all leading and trailing occurrences of a set of specified characters (and ' ', '\t', '\r', '\n') from a string.

```
int Trim(  
    const string targets    // Set of characters to remove  
)
```

### Parameters

*targets*

[in] Set of characters to remove.

### Returned value

Number of characters removed.

### Example:

```
///--- example for CString::Trim  
#include <Strings\String.mqh>  
///---  
void OnStart()  
{  
    CString str;  
    ///---  
    str.Assign(" \t\tABCD\r\n");  
    printf("Source string '%s'", str.Str());  
    ///---  
    str.Trim("DA-DA-DA");  
    printf("Result string '%s'", str.Str());  
}
```

## TrimLeft

Removes all leading occurrences of a set of specified characters (and ' ', '\t', '\r', '\n') from a string.

```
int TrimLeft(  
    const string targets    // Set of characters to remove  
)
```

### Parameters

*targets*

[in] Set of characters to remove.

### Returned value

Number of characters removed.

## TrimRight

Removes all trailing occurrences of a set of specified characters (and ' ', '\t', '\r', '\n') from a string.

```
int TrimRight(  
    const string targets    // Set of characters to remove  
)
```

### Parameters

*targets*

[in] Set of characters to remove.

### Returned value

Number of characters removed.

## Clear

Clears a string.

```
bool Clear()
```

### Returned value

true if successful, false if a string hasn't been cleared.

## ToUpper

Converts a string to uppercase.

```
bool ToUpper ()
```

### Returned value

true if successful, false if a string hasn't been converted.

## ToLower

Converts a string to lowercase.

```
bool ToLower ()
```

### Returned value

true if successful, false if a string hasn't been converted.



## Reverse

Reverses of a string.

```
void Reverse ()
```

## Find

Searches for the first match of a substring.

```
int Find(  
    uint      start,          // Position  
    const string substring    // Substring to search for  
) const;
```

### Parameters

*start*

[in] The index of the character in the string to begin the search with, or 0 to start from the beginning.

*substring*

[in] Substring to search for.

### Returned value

The index of the first character that matches the requested substring; -1 if the substring is not found.

## FindRev

Searches for the last match of a substring.

```
int FindRev(  
    const string substring // Substring  
    ) const;
```

### Parameters

*substring*

[in] A substring to search for.

### Returned value

The index of the last character that matches the requested substring; -1 if the substring is not found.

## Remove

Deletes a substring from a string.

```
uint Remove(  
    const string substring    // Substring to remove  
)
```

### Parameters

*substring*

[in] A substring to search for.

### Returned value

Number of substrings deleted.

## Replace

Replaces a substring from a string.

```
uint Replace(  
    const string  substring,    // Substring to replace  
    const string  newstring     // New substring  
)
```

### Parameters

*substring*

[in] A substring to search for.

*newstring*

[in] A substring to replace for.

### Returned value

Number of substrings replaced.

## Technical Indicators and Timeseries

This section contains the technical details of the technical indicator and timeseries classes and description of the corresponding components of the standard MQL5 library.

The use of the technical indicator and timeseries classes will save time in developing applications (scripts, Expert Advisors).

The MQL5 Standard Library is placed in the working directory of the terminal in the Include\Indicators folder.

| Class/group                              | Description                           |
|------------------------------------------|---------------------------------------|
| <a href="#">Base classes</a>             | Group of base and auxiliary classes   |
| <a href="#">Timeseries classes</a>       | Group of timeseries classes           |
| <a href="#">Trend Indicators</a>         | Group of Trend indicator classes      |
| <a href="#">Oscillators</a>              | Group of Oscillator indicator classes |
| <a href="#">Volume Indicators</a>        | Group of Volume indicator classes     |
| <a href="#">Bill Williams Indicators</a> | Bill Williams indicator classes       |
| <a href="#">Custom indicators</a>        | Custom indicator class                |

## Base and Auxiliary Technical Indicator and Timeseries Classes

This section contains the technical details of base and auxiliary technical indicator and timeseries classes and description of the corresponding components of the Standard MQL5 library.

| Class/group                              | Description                               |
|------------------------------------------|-------------------------------------------|
| <a href="#"><u>CSpreadBuffer</u></a>     | Historical spread buffer class            |
| <a href="#"><u>CTimeBuffer</u></a>       | Historical opening prices buffer class    |
| <a href="#"><u>CTickVolumeBuffer</u></a> | Historical tick volumes buffer class      |
| <a href="#"><u>CRealVolumeBuffer</u></a> | Historical real volumes buffer class      |
| <a href="#"><u>CDoubleBuffer</u></a>     | Base class of double type data buffer     |
| <a href="#"><u>COpenBuffer</u></a>       | Opening bar prices buffer class           |
| <a href="#"><u>CHighBuffer</u></a>       | High bar prices buffer class              |
| <a href="#"><u>CLowBuffer</u></a>        | Low bar prices buffer class               |
| <a href="#"><u>CCloseBuffer</u></a>      | Closing bar prices buffer class           |
| <a href="#"><u>CIndicatorBuffer</u></a>  | Technical indicator buffer class          |
| <a href="#"><u>CSeries</u></a>           | Base class for access to timeseries data  |
| <a href="#"><u>CPriceSeries</u></a>      | Base class for access to price data       |
| <a href="#"><u>CIndicator</u></a>        | Base class of technical indicator         |
| <a href="#"><u>Indicators</u></a>        | Technical indicator and series collection |

## CSpreadBuffer

CSpreadBuffer is a class for simplified access to spreads of the bars in the history.

### Description

The CSpreadBuffer class provides an access to spreads of the bars in the history.

### Declaration

```
class CSpreadBuffer: public CArrayInt
```

### Title

```
#include <Indicators\TimeSeries.mqh>
```

### Class Methods

| Attributes                             |                                  |
|----------------------------------------|----------------------------------|
| <a href="#">Size</a>                   | Sets buffer size                 |
| Settings                               |                                  |
| <a href="#">SetSymbolPeriod</a>        | Sets symbol and period           |
| Data Access Methods                    |                                  |
| <a href="#">At</a>                     | Gets the buffer element by index |
| Data Update Methods                    |                                  |
| virtual <a href="#">Refresh</a>        | Updates the buffer               |
| virtual <a href="#">RefreshCurrent</a> | Updates the current value        |



## Size

Sets buffer size.

```
void Size(  
    const int size    // new size  
)
```

### Parameters

*size*

[in] New buffer size.

## SetSymbolPeriod

Sets symbol and period.

```
void SetSymbolPeriod(  
    const string      symbol,      // symbol  
    const ENUM_TIMEFRAMES period    // period  
)
```

### Parameters

*symbol*

[in] New symbol.

*period*

[in] New period ([ENUM\\_TIMEFRAMES](#) enumeration).

## At

Gets the buffer element by index.

```
int At(  
    const int index    // index  
) const
```

### Parameters

*index*

[in] Index of buffer element.

### Returned value

Buffer element with the specified index.

## Refresh

Updates the buffer.

```
virtual bool Refresh()
```

### Returned value

true - if successful, false - if the buffer hasn't been updated.

## RefreshCurrent

Updates the current (zeroth) element of the buffer.

```
virtual bool RefreshCurrent()
```

### Returned value

true - if successful, false - if the buffer hasn't been updated.

## CTimeBuffer

CTimeBuffer is a class for simplified access to opening times of the bars in the history.

### Description

The CTimeBuffer class provides an access to opening times of the bars in the history.

### Declaration

```
class CTimeBuffer: public CArrayLong
```

### Title

```
#include <Indicators\TimeSeries.mqh>
```

### Class Methods

| Attributes                             |                                  |
|----------------------------------------|----------------------------------|
| <a href="#">Size</a>                   | Sets buffer size                 |
| Settings                               |                                  |
| <a href="#">SetSymbolPeriod</a>        | Sets symbol and period           |
| Data Access Methods                    |                                  |
| <a href="#">At</a>                     | Gets the buffer element by index |
| Data Update Methods                    |                                  |
| virtual <a href="#">Refresh</a>        | Updates the buffer               |
| virtual <a href="#">RefreshCurrent</a> | Updates the current value        |

## Size

Sets buffer size.

```
void Size(  
    const int size    // new size  
)
```

### Parameters

*size*

[in] New buffer size.

## SetSymbolPeriod

Sets symbol and period.

```
void SetSymbolPeriod(  
    const string      symbol,      // symbol  
    const ENUM_TIMEFRAMES period    // period  
)
```

### Parameters

*symbol*

[in] New symbol.

*period*

[in] New period ([ENUM\\_TIMEFRAMES](#) enumeration).



## At

Gets the buffer element by index.

```
long At(  
    const int index    // index  
) const
```

### Parameters

*index*

[in] Index of buffer element.

### Returned value

Buffer element with the specified index.

## Refresh

Updates the buffer.

```
virtual bool Refresh()
```

### Returned value

true - if successful, false - if the buffer hasn't been updated.

## RefreshCurrent

Updates the current (zeroth) element of the buffer.

```
virtual bool RefreshCurrent()
```

### Returned value

true - if successful, false - if the buffer hasn't been updated.

## CTickVolumeBuffer

CTickVolumeBuffer is a class for simplified access to tick volumes of bars in the history.

### Description

The CTickVolumeBuffer class provides an access to tick volumes of bars in the history.

### Declaration

```
class CTickVolumeBuffer: public CArrayLong
```

### Title

```
#include <Indicators\TimeSeries.mqh>
```

### Class Methods

| Attributes                             |                                  |
|----------------------------------------|----------------------------------|
| <a href="#">Size</a>                   | Sets buffer size                 |
| Settings                               |                                  |
| <a href="#">SetSymbolPeriod</a>        | Sets symbol and period           |
| Data Access Methods                    |                                  |
| <a href="#">At</a>                     | Gets the buffer element by index |
| Data Update Methods                    |                                  |
| virtual <a href="#">Refresh</a>        | Updates the buffer               |
| virtual <a href="#">RefreshCurrent</a> | Updates the current value        |

## Size

Sets buffer size.

```
void Size(  
    const int size    // new size  
)
```

### Parameters

*size*

[in] New buffer size.

## SetSymbolPeriod

Sets symbol and period.

```
void SetSymbolPeriod(  
    const string      symbol,      // symbol  
    const ENUM_TIMEFRAMES period    // period  
)
```

### Parameters

*symbol*

[in] New symbol.

*period*

[in] New period ([ENUM\\_TIMEFRAMES](#) enumeration).

## At

Gets the buffer element by index.

```
long At(  
    const int index    // index  
) const
```

### Parameters

*index*

[in] Index of buffer element.

### Returned value

Buffer element with the specified index.

## Refresh

Updates the buffer.

```
virtual bool Refresh()
```

### Returned value

true - if successful, false - if the buffer hasn't been updated.



## RefreshCurrent

Updates the current (zeroth) element of the buffer.

```
virtual bool RefreshCurrent()
```

### Returned value

true - if successful, false - if the buffer hasn't been updated.

## CRealVolumeBuffer

CRealVolumeBuffer is a class for simplified access to real volumes of bars in the history.

### Description

The CRealVolumeBuffer class provides an access to real volumes of bars in the history.

### Declaration

```
class CRealVolumeBuffer: public CArrayLong
```

### Title

```
#include <Indicators\TimeSeries.mqh>
```

### Class Methods

| Attributes                             |                                  |
|----------------------------------------|----------------------------------|
| <a href="#">Size</a>                   | Sets buffer size                 |
| Settings                               |                                  |
| <a href="#">SetSymbolPeriod</a>        | Sets symbol and period           |
| Data Access Methods                    |                                  |
| <a href="#">At</a>                     | Gets the buffer element by index |
| Data Update Methods                    |                                  |
| virtual <a href="#">Refresh</a>        | Updates the buffer               |
| virtual <a href="#">RefreshCurrent</a> | Updates the current value        |

## Size

Sets buffer size.

```
void Size(  
    const int size    // new size  
)
```

### Parameters

*size*

[in] New buffer size.

## SetSymbolPeriod

Sets symbol and period.

```
void SetSymbolPeriod(  
    const string      symbol,      // symbol  
    const ENUM_TIMEFRAMES period    // period  
)
```

### Parameters

*symbol*

[in] New symbol.

*period*

[in] New period ([ENUM\\_TIMEFRAMES](#) enumeration).

## At

Gets the buffer element by index.

```
long At(  
    const int index    // index  
) const
```

### Parameters

*index*

[in] Index of buffer element.

### Returned value

Buffer element with the specified index.

## Refresh

Updates the buffer.

```
virtual bool Refresh()
```

### Returned value

true - if successful, false - if the buffer hasn't been updated.

## RefreshCurrent

Updates the current (zeroth) element of the buffer.

```
virtual bool RefreshCurrent()
```

### Returned value

true - if successful, false - if the buffer hasn't been updated.

## CDoubleBuffer

CDoubleBuffer is a base class for simplified access to data buffers of double type.

### Description

The CDoubleBuffer class provides an access to the data of the buffer of double type.

### Declaration

```
class CDoubleBuffer: public CArrayDouble
```

### Title

```
#include <Indicators\TimeSeries.mqh>
```

### Class Methods

| Attributes                             |                           |
|----------------------------------------|---------------------------|
| <a href="#">Size</a>                   | Sets buffer size          |
| Settings                               |                           |
| <a href="#">SetSymbolPeriod</a>        | Sets symbol and period    |
| Data Access Methods                    |                           |
| <a href="#">At</a>                     | Gets the buffer element   |
| Data Update Methods                    |                           |
| virtual <a href="#">Refresh</a>        | Updates the buffer        |
| virtual <a href="#">RefreshCurrent</a> | Updates the current value |



## Size

Sets buffer size.

```
void Size(  
    const int size    // new size  
)
```

### Parameters

*size*

[in] New buffer size.

## SetSymbolPeriod

Sets symbol and period.

```
void SetSymbolPeriod(  
    const string      symbol,      // symbol  
    const ENUM_TIMEFRAMES period    // period  
)
```

### Parameters

*symbol*

[in] New symbol.

*period*

[in] New period ([ENUM\\_TIMEFRAMES](#) enumeration).

## At

Gets the buffer element by index.

```
double At(  
    const int index    // index  
) const
```

### Parameters

*index*

[in] Index of buffer element.

### Returned value

Buffer element with the specified index.

## Refresh

Updates the buffer.

```
virtual bool Refresh()
```

### Returned value

true - if successful, false - if the buffer hasn't been updated.

## RefreshCurrent

Updates the current (zeroth) element of the buffer.

```
virtual bool RefreshCurrent()
```

### Returned value

true - if successful, false - if the buffer hasn't been updated.

## COpenBuffer

COpenBuffer is a class for simplified access to open prices of bars in the history.

### Description

The COpenBuffer class provides an access to open prices of bars in the history.

### Declaration

```
class COpenBuffer: public CDoubleBuffer
```

### Title

```
#include <Indicators\TimeSeries.mqh>
```

### Class Methods

| Data Update Methods                    |                           |
|----------------------------------------|---------------------------|
| virtual <a href="#">Refresh</a>        | Updates the buffer        |
| virtual <a href="#">RefreshCurrent</a> | Updates the current value |

## Refresh

Updates the buffer.

```
virtual bool Refresh()
```

### Returned value

true - if successful, false - if the buffer hasn't been updated.

## RefreshCurrent

Updates the current (zeroth) element of the buffer.

```
virtual bool RefreshCurrent()
```

### Returned value

true - if successful, false - if the buffer hasn't been updated.



## CHighBuffer

CHighBuffer is a class for simplified access to high prices of bars in the history.

### Description

The CHighBuffer class provides an access to high prices of bars in the history.

### Declaration

```
class CHighBuffer: public CDoubleBuffer
```

### Title

```
#include <Indicators\TimeSeries.mqh>
```

### Class Methods

| Data Update Methods                    |                           |
|----------------------------------------|---------------------------|
| virtual <a href="#">Refresh</a>        | Updates the buffer        |
| virtual <a href="#">RefreshCurrent</a> | Updates the current value |

## Refresh

Updates the buffer.

```
virtual bool Refresh()
```

### Returned value

true - if successful, false - if the buffer hasn't been updated.

## RefreshCurrent

Updates the current (zeroth) element of the buffer.

```
virtual bool RefreshCurrent()
```

### Returned value

true - if successful, false - if the buffer hasn't been updated.

## CLowBuffer

CLowBuffer is a class for simplified access to low prices of bars in the history.

### Description

The CLowBuffer class provides an access to low prices of bars in the history.

### Declaration

```
class CLowBuffer: public CDoubleBuffer
```

### Title

```
#include <Indicators\TimeSeries.mqh>
```

### Class Methods

| Data Update Methods                    |                           |
|----------------------------------------|---------------------------|
| virtual <a href="#">Refresh</a>        | Updates the buffer        |
| virtual <a href="#">RefreshCurrent</a> | Updates the current value |

## Refresh

Updates the buffer.

```
virtual bool Refresh()
```

### Returned value

true - if successful, false - if the buffer hasn't been updated.

## RefreshCurrent

Updates the current (zeroth) element of the buffer.

```
virtual bool RefreshCurrent()
```

### Returned value

true - if successful, false - if the buffer hasn't been updated.

## CCloseBuffer

CCloseBuffer is a class for simplified access to close prices of bars in the history.

### Description

The CCloseBuffer class provides an access to close prices of bars in the history.

### Declaration

```
class CCloseBuffer: public CDoubleBuffer
```

### Title

```
#include <Indicators\TimeSeries.mqh>
```

### Class Methods

| Data Update Methods                    |                           |
|----------------------------------------|---------------------------|
| virtual <a href="#">Refresh</a>        | Updates the buffer        |
| virtual <a href="#">RefreshCurrent</a> | Updates the current value |

## Refresh

Updates the buffer.

```
virtual bool Refresh()
```

### Returned value

true - if successful, false - if the buffer hasn't been updated.



## RefreshCurrent

Updates the current (zeroth) element of the buffer.

```
virtual bool RefreshCurrent()
```

### Returned value

true - if successful, false - if the buffer hasn't been updated.

## CIndicatorBuffer

CIndicatorBuffer is a class for simplified access to the data of the indicator's buffer.

### Description

The CIndicatorBuffer class provides the simplified access to the data buffer of technical indicator.

### Declaration

```
class CIndicatorBuffer: public CDoubleBuffer
```

### Title

```
#include <Indicators\Indicator.mqh>
```

### Class Methods

| Attributes                     |                                |
|--------------------------------|--------------------------------|
| <a href="#">Offset</a>         | Gets/Sets offset of the buffer |
| <a href="#">Name</a>           | Gets/Sets buffer name          |
| Data Access Methods            |                                |
| <a href="#">At</a>             | Gets buffer's element          |
| Data Update Methods            |                                |
| <a href="#">Refresh</a>        | Updates the buffer             |
| <a href="#">RefreshCurrent</a> | Updates only current value     |

## Offset

Gets offset of the buffer.

```
int Offset() const
```

### Returned value

Buffer offset.

## Offset

Sets offset of the buffer.

```
void Offset(  
    const int offset    // offset  
)
```

### Parameters

*offset*

[in] New buffer offset.

## Name

Gets the name of the buffer.

```
string Name() const
```

### Returned value

Name of the buffer.

## Name

Sets the name of the buffer.

```
void Name(  
    const string name    // name  
)
```

### Parameters

*name*

[in] New name of the buffer.

## At

Gets buffer element by index.

```
double At(  
    int index    // index  
) const
```

### Parameters

*index*

[in] Index of buffer element.

### Returned value

Buffer element with the specified index.

## Refresh

Updates the whole buffer.

```
bool Refresh(  
    const int handle,    // handle  
    const int num        // buffer number  
)
```

### Parameters

*handle*

[in] Handle of the indicator.

*num*

[in] Buffer index of the indicator.

### Returned value

true - if successful, false - if the buffer hasn't been updated.

## RefreshCurrent

Updates the current (zeroth) buffer element.

```
bool RefreshCurrent(  
    const int handle,    // handle of the indicator  
    const int num        // buffer number  
)
```

### Parameters

*handle*

[in] Handle of the indicator.

*num*

[in] Buffer number.

### Returned value

true - if successful, false - if the buffer hasn't been updated.

## CSeries

CSeries is a base class for an access to the timeseries data of the Standard Library.

### Description

The CSeries class provides the simplified access to MQL5 timeseries functions to all its descendants.

### Declaration

```
class CSeries: public CArrayObj
```

### Title

```
#include <Indicators\Series.mqh>
```

### Class Methods

| Attributes                  |                                                       |
|-----------------------------|-------------------------------------------------------|
| <u>Name</u>                 | Gets the name of timeseries or indicator              |
| <u>BuffersTotal</u>         | Gets the number of buffers of timeseries or indicator |
| <u>Timeframe</u>            | Gets the timeframe flag of timeseries or indicator    |
| <u>Symbol</u>               | Gets the symbol of timeseries or indicator            |
| <u>Period</u>               | Gets the period of timeseries or indicator            |
| <u>RefreshCurrent</u>       | Gets/Sets the flag of updating the current data       |
| <b>Data Access Methods</b>  |                                                       |
| virtual <u>BufferResize</u> | Sets buffer size of timeseries or indicator           |
| <b>Data Update Methods</b>  |                                                       |
| virtual <u>Refresh</u>      | Update the data of timeseries or indicator            |
| <u>PeriodDescription</u>    | Gets the period as a string                           |



## Name

Gets the name of timeseries or indicator

```
string Name() const
```

### Returned value

The name of timeseries or indicator.

## BuffersTotal

Gets the number of buffers of timeseries or indicator.

```
int BuffersTotal() const
```

### Returned value

The number of buffers of timeseries or indicator.

### Note

The timeseries has an only one buffer.

## Timeframe

Gets the timeframe flag of timeseries or indicator.

```
int Timeframe() const
```

### Returned value

The timeframe flag of timeseries or indicator.

### Note

It's the visibility flag of some timeframes.

## Symbol

Gets the symbol of timeseries or indicator.

```
string Symbol() const
```

### Returned value

The symbol of timeseries or indicator.

## Period

Gets the period of timeseries or indicator.

```
ENUM_TIMEFRAMES Period() const
```

### Returned value

The period (value of [ENUM\\_TIMEFRAMES](#) enumeration) of timeseries or indicator.

## RefreshCurrent

Sets a flag to update the current values of timeseries or indicator.

```
string RefreshCurrent(  
    const bool flag    // new flag  
)
```

### Parameters

*flag*

[in] New flag.

### Returned value

None.

## BufferResize

Sets buffer size of timeseries or indicator.

```
virtual bool BufferResize(  
    const int size    // new size  
)
```

### Parameters

*size*

[in] New size of the buffers.

### Returned value

true - if successful, false - otherwise.

### Note

All the timeseries or indicator buffers have the same size.

## Refresh

Updates the data of timeseries or indicator.

```
virtual void Refresh(  
    const int flags    // flags  
)
```

### Parameters

*flags*

[in] Timeframes to update (flag).



## PeriodDescription

Gets the string representation of the specified [ENUM\\_TIMEFRAMES](#) enumeration.

```
string PeriodDescription(  
    const int val=0      // value  
)
```

### Parameters

*val=0*

[in] Value to convert.

### Returned value

The string representation of the specified [ENUM\\_TIMEFRAMES](#) enumeration.

### Note

If the value isn't specified or equal to zero, it returns the timeframe of timeseries or indicator.

## CPriceSeries

CPriceSeries is a base class for access to the price data.

### Description

The CSeries class provides the simplified access to MQL5 functions for working with price data to all its descendants.

### Declaration

```
class CPriceSeries: public CSeries
```

### Title

```
#include <Indicators\TimeSeries.mqh>
```

### Class Methods

| Create Methods                       |                                                                    |
|--------------------------------------|--------------------------------------------------------------------|
| virtual <a href="#">BufferResize</a> | Sets size of the buffer                                            |
| Data Access Methods                  |                                                                    |
| virtual <a href="#">GetData</a>      | Gets the specified buffer element by index                         |
| Data Update Methods                  |                                                                    |
| virtual <a href="#">Refresh</a>      | Updates timeseries data                                            |
| Data Search Methods                  |                                                                    |
| virtual <a href="#">MinIndex</a>     | Gets the index of minimal element in the specified range           |
| virtual <a href="#">MinValue</a>     | Gets the value and index of minimal element in the specified range |
| virtual <a href="#">MaxIndex</a>     | Gets the index of maximal element in the specified range           |
| virtual <a href="#">MaxValue</a>     | Gets the value and index of maximal element in the specified range |

## BufferResize

Sets new size of the buffer.

```
virtual void BufferResize(  
    const int size    // new size  
)
```

### Parameters

*size*

[in] New buffer size.

## GetData

Gets the specified buffer element by index.

```
double GetData(  
    const int index    // index  
) const
```

### Parameters

*index*

[in] Index of buffer element.

### Returned value

The buffer element with the specified index or [EMPTY\\_VALUE](#).

## Refresh

Updates the timeseries data

```
virtual void Refresh(  
    const int flags=OBJ_ALL_PERIODS    // timeframe flags  
)
```

### Parameters

*flags=OBJ\_ALL\_PERIODS*

[in] Timeframes to update (flag).

## MinIndex

Gets the index of minimal element in the specified range.

```
virtual int MinIndex(  
    const int start,      // starting index  
    const int count       // number of elements to scan  
) const
```

### Parameters

*start*

[in] Starting index.

*count*

[in] Number of elements to proceed.

### Returned value

The index of minimal element in the specified range, or -1 in the case of error.

## MinValue

Gets the value and index of minimal element in the specified range.

```
virtual double MinValue(  
    const int    start,      // starting index  
    const int    count,     // number of elements to scan  
    int&         index       // reference to the variable for index  
) const
```

### Parameters

*start*

[in] Starting index.

*count*

[in] Number of elements to proceed.

*index*

[out] Reference to the variable of integer type.

### Returned value

The value of minimal element of the buffer in the specified range, or [EMPTY\\_VALUE](#) if error.

### Note

The index of minimal element is stored in the variable index.

## MaxIndex

Gets the index of maximal element in the specified range.

```
virtual int MaxIndex(  
    const int start,      // starting index  
    const int count       // number of elements to scan  
) const
```

### Parameters

*start*

[in] Starting index.

*count*

[in] Number of elements to proceed.

### Returned value

The index of the maximal element in the specified range, or -1 in the case of error.



## MaxValue

Gets the value and index of maximal element in the specified range.

```
virtual double MaxValue(  
    const int start,      // starting index  
    const int count,      // number of elements to scan  
    int& index            // reference to the variable for index  
) const
```

### Parameters

*start*

[in] Starting index.

*count*

[in] Number of elements to proceed.

*index*

[out] Reference to the variable of integer type.

### Returned value

The value of maximal element of the buffer in the specified range, or [EMPTY\\_VALUE](#) if error.

### Note

The index of maximal element is stored in the variable index.

## CIndicator

CIndicator is a base class for technical indicator classes of the standard MQL Library.

### Description

The CIndicator class provides the simplified access for all of its descendants to MQL5 API technical indicator functions.

### Declaration

```
class CIndicator: public CSeries
```

### Title

```
#include <Indicators\Indicator.mqh>
```

### Class Methods

| Attributes                        |                                                                                           |
|-----------------------------------|-------------------------------------------------------------------------------------------|
| <a href="#">Handle</a>            | Gets the indicator's handle.                                                              |
| <a href="#">Status</a>            | Gets the status of the indicator.                                                         |
| <a href="#">FullRelease</a>       | Sets a flag to release the handle of the indicator.                                       |
| <b>Creation</b>                   |                                                                                           |
| <a href="#">Create</a>            | Creates the indicators                                                                    |
| <a href="#">BufferResize</a>      | Sets new buffer size                                                                      |
| <b>Data Access Methods</b>        |                                                                                           |
| <a href="#">GetData</a>           | Copying the data from the indicator's buffer                                              |
| <b>Data Update Methods</b>        |                                                                                           |
| <a href="#">Refresh</a>           | Updates the indicator's data                                                              |
| <b>Finding Min/Max Values</b>     |                                                                                           |
| <a href="#">Minimum</a>           | Gets the index of minimal element of the specified buffer in a specified range.           |
| <a href="#">MinValue</a>          | Gets the value and index of minimal element of the specified buffer in a specified range. |
| <a href="#">Maximum</a>           | Gets the index of maximal element of the specified buffer in a specified range.           |
| <a href="#">MaxValue</a>          | Gets the value and index of maximal element of the specified buffer in a specified range. |
| <b>Conversion of Enumerations</b> |                                                                                           |

|                                          |                                                             |
|------------------------------------------|-------------------------------------------------------------|
| <a href="#"><u>MethodDescription</u></a> | Gets the value of ENUM_MA_METHOD enumeration as string      |
| <a href="#"><u>PriceDescription</u></a>  | Gets the value of ENUM_APPLIED_PRICE enumeration as string  |
| <a href="#"><u>VolumeDescription</u></a> | Gets the value of ENUM_APPLIED_VOLUME enumeration as string |
| <b>Working with chart</b>                |                                                             |
| <a href="#"><u>AddToChart</u></a>        | Adds the indicator to the chart                             |
| <a href="#"><u>DeleteFromChart</u></a>   | Deletes the indicator from the chart                        |

**Derived classes:**

- [CiAC](#)
- [CiAD](#)
- [CiADX](#)
- [CiADXWilder](#)
- [CiAlligator](#)
- [CiAMA](#)
- [CiAO](#)
- [CiATR](#)
- [CiBands](#)
- [CiBearsPower](#)
- [CiBullsPower](#)
- [CiBWMFI](#)
- [CiCCI](#)
- [CiChaikin](#)
- [CiDEMA](#)
- [CiDeMarker](#)
- [CiEnvelopes](#)
- [CiForce](#)
- [CiFractals](#)
- [CiFrAMA](#)
- [CiGator](#)
- [CiIchimoku](#)
- [CiMA](#)
- [CiMACD](#)
- [CiMFI](#)
- [CiMomentum](#)
- [CiOBV](#)

- [CiOsMA](#)
- [CiRSI](#)
- [CiRVI](#)
- [CiSAR](#)
- [CiStdDev](#)
- [CiStochastic](#)
- [CiTEMA](#)
- [CiTriX](#)
- [CiVIDyA](#)
- [CiVolumes](#)
- [CiWPR](#)

## Handle

Gets the indicator's handle.

```
int Handle() const
```

### Returned value

Handle of the indicator.

## Status

Gets the status of the indicator.

```
string Status() const
```

### Returned value

The status of indicator creation.

## FullRelease

Sets a flag to release the handle of the indicator.

```
void FullRelease(  
    const bool flag=true    // flag  
)
```

### Parameters

*flag*

[in] New value of the handle release flag.

## Create

Creates the indicator with specified parameters.

```
bool Create(  
    const string      symbol,          // symbol  
    const ENUM_TIMEFRAMES period,      // period  
    const ENUM_INDICATOR type,         // type  
    const int         num_params,      // number of parameters  
    const MqlParam&   params[]        // reference to the parameters array  
)
```

### Parameters

*symbol*

[in] Symbol name.

*period*

[in] Period ([ENUM\\_TIMEFRAMES](#) enumeration).

*type*

[in] Indicator's type ([ENUM\\_INDICATOR](#) enumeration).

*num\_params*

[in] Number of indicator's parameters.

*params*

[in] Reference to the parameters array for the indicator.

### Returned value

true if successful, false if indicator hasn't been created.



## BufferResize

Sets the size of the indicator's buffer.

```
virtual bool BufferResize(  
    const int size    // size  
)
```

### Parameters

*size*

[in] New buffer size.

### Returned value

true - if successful, false - otherwise.

### Note

All the indicator's buffers have the same size.

## BarsCalculated

Returns the number of calculated data for the indicator.

```
int BarsCalculated() const;
```

### Return Value

Returns the amount of calculated data in the indicator buffer or -1 in the case of error (data not calculated yet).

## GetData

Gets the specified element from the specified buffer of the indicator. [Refresh\(\)](#) should be called for working with recent data before using the method.

```
double  GetData (
    const int  buffer_num,      // buffer number
    const int  index           // element index
) const
```

### Parameters

*buffer\_num*  
[in] Buffer number.

*index*  
[in] Element index.

### Returned value

If successful, it returns the numerical value of element, or [EMPTY\\_VALUE](#) in the case of error.

## GetData

Gets the data from the indicator's buffer by starting position and number of necessary data.

```
int  GetData (
    const int      start_pos,      // position
    const int      count,         // number of elements needed
    const int      buffer_num,     // buffer number
    double&        buffer[]       // target array for data
) const
```

### Parameters

*start\_pos*  
[in] Starting position of the indicator's buffer.

*count*  
[in] Number of elements needed.

*buffer\_num*  
[in] Number of the indicator's buffer.

*buffer*  
[in] Reference to the target array for the data.

### Returned value

If successful, it returns the number of elements, received from the specified indicator buffer, otherwise -1.

## GetData

Gets the data from the indicator's buffer by start time and number of necessary data.

```
int GetData(  
    const datetime start_time, // starting time  
    const int count, // number of elements needed  
    const int buffer_num, // buffer number  
    double& buffer[] // target array for data  
) const
```

#### Parameters

*start\_time*  
[in] Starting time.

*count*  
[in] Number of elements needed.

*buffer\_num*  
[in] Number of the indicator's buffer.

*buffer*  
[in] Reference to the target array.

#### Returned value

If successful, it returns the number of elements, received from the specified indicator buffer, otherwise -1.

## GetData

Gets the data from the indicator's buffer by start and stop time and number of necessary data.

```
int GetData(  
    const datetime start_time, // start time  
    const datetime stop_time, // stop time  
    const int buffer_num, // number of buffer  
    double& buffer[] // target array for data  
) const
```

#### Parameters

*start\_time*  
[in] Starting time.

*stop\_time*  
[in] Stop time.

*buffer\_num*  
[in] Number of the indicator's buffer.

*buffer*  
[in] Reference to the target array.

**Returned value**

If successful, it returns the number of elements, received from the specified indicator buffer, otherwise -1.

## Refresh

Updates the indicator's data. It is recommended calling the method before using [GetData\(\)](#).

```
virtual void Refresh(  
    int flags=OBJ_ALL_PERIODS    // flags  
)
```

### Parameters

*flags=OBJ\_ALL\_PERIODS*

[in] Timeframe update flags.

## Minimum

Returns the index of minimal element of the specified buffer in a specified range.

```
int Minimum(  
    const int  buffer_num,      // buffer number  
    const int  start,           // starting index  
    const int  count            // number of elements to proceed  
) const
```

### Parameters

*buffer\_num*

[in] Buffer number to search the value in.

*start*

[in] Starting index of the search.

*count*

[in] Number of elements to search.

### Returned value

Index of the minimal element of the specified buffer in a specified range.

## MinValue

Returns the value and index of minimal element of the specified buffer in a specified range.

```
double MinValue(  
    const int    buffer_num,      // buffer number  
    const int    start,           // starting index  
    const int    count,           // number of elements to proceed  
    int&         index            // reference  
) const
```

### Parameters

*buffer\_num*

[in] Buffer number to search the value in.

*start*

[in] Starting index.

*count*

[in] Number of elements to proceed.

*index*

[out] Reference to the variable of int type for the minimal element index.

### Returned value

The value of the minimal element of the specified buffer in a specified range.

### Note

The index of minimal buffer element is stored into the variable *index*, passed by reference.



## Maximum

Returns the index of maximal element of the specified buffer in a specified range.

```
int Maximum(  
    const int  buffer_num,    // buffer number  
    const int  start,        // starting index  
    const int  count         // number of elements to proceed  
) const
```

### Parameters

*buffer\_num*

[in] Buffer number to search the value in.

*start*

[in] Starting index of the search.

*count*

[in] Number of elements to search.

### Returned value

Index of the maximal element of the specified buffer in a specified range.

## MaxValue

Returns the value and index of maximal element of the specified buffer in a specified range.

```
double MaxValue(  
    const int    buffer_num,      // buffer number  
    const int    start,           // starting index  
    const int    count,           // number of elements to proceed  
    int&         index            // reference  
) const
```

### Parameters

*buffer\_num*

[in] Buffer number to search the value in.

*start*

[in] Starting index.

*count*

[in] Number of elements to proceed.

*index*

[out] Reference to the variable of int type for the maximal element index.

### Returned value

The value of the maximal element of the specified buffer in a specified range.

### Note

The index of maximal buffer element is stored into the variable *index*, passed by reference.

## MethodDescription

The function returns the value of [ENUM\\_MA\\_METHOD](#) enumeration as a string.

```
string MethodDescription(  
    const int val    // value  
) const
```

### Parameters

*val*

[in] Value of [ENUM\\_MA\\_METHOD](#) enumeration.

### Returned value

The value of [ENUM\\_MA\\_METHOD](#) enumeration as a string.

## PriceDescription

The method returns the value of [ENUM\\_APPLIED\\_PRICE](#) enumeration as a string.

```
string PriceDescription(  
    const int val    // value  
) const
```

### Parameters

*val*

[in] Value of [ENUM\\_APPLIED\\_PRICE](#) enumeration.

### Returned value

The value of [ENUM\\_APPLIED\\_PRICE](#) enumeration as a string.

## VolumeDescription

The method returns the value of [ENUM\\_APPLIED\\_VOLUME](#) enumeration as a string.

```
string VolumeDescription(  
    const int val    // value  
) const
```

### Parameters

*val*

[in] Value of [ENUM\\_APPLIED\\_VOLUME](#) enumeration.

### Returned value

The value of [ENUM\\_APPLIED\\_VOLUME](#) enumeration as a string.

## AddToChart

Adds the indicator to the chart.

```
bool AddToChart(  
    const long chart,      // chart ID  
    const int subwin       // chart subwindow  
)
```

### Parameters

*chart*

[in] Chart ID.

*subwin*

[in] Chart subwindow.

### Returned value

true - if successful, false if error.

## DeleteFromChart

Deletes the indicator from the chart.

```
bool DeleteFromChart(  
    const long chart,      // chart ID  
    const int subwin       // chart subwindow  
)
```

### Parameters

*chart*

[in] Chart ID.

*subwin*

[in] Chart subwindow.

### Returned value

true - if successful, false if error.

## Clndicators

The Clndicators is a class for collecting instances of timeseries and technical indicators classes.

### Description

The Clndicators class provides creation of the technical indicators class instances, their storage and management (data synchronization, handle and memory management).

### Declaration

```
class CIndicators: public CArrayObj
```

### Title

```
#include <Indicators\Indicators.mqh>
```

### Class Methods

| Create Methods          |                                                             |
|-------------------------|-------------------------------------------------------------|
| <a href="#">Create</a>  | Creates technical indicator                                 |
| Data Update Methods     |                                                             |
| <a href="#">Refresh</a> | Updates data for all technical indicators in the collection |



## Create

It creates the indicator with the specified parameters.

```
CIndicator* Create(  
    const string          symbol,      // Symbol name  
    const ENUM_TIMEFRAMES period,      // Period  
    const ENUM_INDICATOR  type,        // Indicator's type  
    const int             count,       // Number of parameters  
    const MqlParam&       params       // Parameters array reference  
)
```

### Parameters

*symbol*

[in] Symbol name.

*period*

[in] Period ([ENUM\\_TIMEFRAMES](#) enumeration).

*type*

[in] Indicator's type ([ENUM\\_INDICATOR](#)).

*count*

[in] Number of parameters for the indicator.

*params*

[in] Reference to the parameters array for the indicator.

### Returned value

If successful, it returns the reference to the created indicator, and NULL if the indicator hasn't been created.

## Refresh

Updates data for all technical indicators in the collection.

```
int Refresh()
```

### Returned value

It returns the updated timeframe flags (formed as an object visibility flags).

## Timeseries classes

This group of chapters contains technical details of timeseries classes of the MQL5 Standard Library and descriptions of all its key components.

| Class                               | Description                                                   |
|-------------------------------------|---------------------------------------------------------------|
| <a href="#"><u>CiSpread</u></a>     | Provides an access to spread historical data                  |
| <a href="#"><u>CiTime</u></a>       | Provides an access to open times of the bars in the history   |
| <a href="#"><u>CiTickVolume</u></a> | Provides an access to tick volumes of the bars in the history |
| <a href="#"><u>CiRealVolume</u></a> | Provides an access to real volumes of the bars in the history |
| <a href="#"><u>CiOpen</u></a>       | Provides an access to open prices of the bars in the history  |
| <a href="#"><u>CiHigh</u></a>       | Provides an access to high prices of the bars in the history  |
| <a href="#"><u>CiLow</u></a>        | Provides an access to low prices of the bars in the history   |
| <a href="#"><u>CiClose</u></a>      | Provides an access to close prices of the bars in the history |

## CiSpread

CiSpread is a class designed for access to spreads of the bars in the history.

### Description

The CiSpread class provides an access to spread historical data.

### Declaration

```
class CiSpread: public CSeries
```

### Title

```
#include <Indicators\TimeSeries.mqh>
```

### Class Methods

| Create Methods               |                      |
|------------------------------|----------------------|
| <a href="#">Create</a>       | Creates a timeseries |
| <a href="#">BufferResize</a> | Sets buffer size     |
| Data Access Methods          |                      |
| <a href="#">GetData</a>      | Gets the data        |
| Data Update Methods          |                      |
| <a href="#">Refresh</a>      | Updates the data     |

## Create

Creates a timeseries with the specified parameters for access to the spreads history.

```
bool Create(  
    string      symbol,      // symbol  
    ENUM_TIMEFRAMES period    // period  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

### Returned value

true if successful, false if timeseries hasn't been created.

## BufferResize

Sets new size of the series.

```
virtual void BufferResize(  
    int size    // new size  
)
```

### Parameters

*size*

[in] New buffer size.

## GetData

Gets the element of timeseries by index.

```
int  GetData(  
    int  index      // index  
    ) const
```

### Parameters

*index*

[in] Index of the element needed.

### Returned value

The timeseries buffer element, or 0.

## GetData

Gets the element of timeseries by starting position and number of elements.

```
int  GetData(  
    int  start_pos,      // starting position  
    int  count,          // number of elements to get  
    int& buffer          // target array  
    ) const
```

### Parameters

*start\_pos*

[in] Starting position of timeseries.

*count*

[in] Number of elements needed.

*buffer*

[in] Reference to the target array for the data.

### Returned value

>=0 if successful, -1 in the case of error.

## GetData

Gets the element of timeseries by starting time and number of elements.

```
int  GetData(  
    datetime start_time,      // starting time  
    int      count,          // number of elements  
    int&     buffer          // target array  
    ) const
```

### Parameters

*start\_time*

[in] Starting time.

*count*

[in] Number of elements needed.

*buffer*

[in] Reference to the target array for data.

#### Returned value

>=0 if successful, -1 in the case of error.

## GetData

Gets the element of timeseries by starting and stop times.

```
int  GetData(  
    datetime  start_time,    // starting time  
    datetime  stop_time,     // stop time  
    int&      buffer         // target array  
) const
```

#### Parameters

*start\_time*

[in] Starting time.

*stop\_time*

[in] Stop time.

*buffer*

[in] Reference to the target array for data

#### Returned value

>=0 if successful, -1 in the case of error.



## Refresh

Updates the data of timeseries.

```
virtual void Refresh(  
    int flags    // flags  
)
```

### Parameters

*flags*

[in] Timeframe flags.

## CiTime

CiTime is a class designed for access to open times of the bars in the history.

### Description

The CiTime class provides an access to open times of the bars in the history.

### Declaration

```
class CiTime: public CSeries
```

### Title

```
#include <Indicators\TimeSeries.mqh>
```

### Class Methods

| Create Methods               |                      |
|------------------------------|----------------------|
| <a href="#">Create</a>       | Creates a timeseries |
| <a href="#">BufferResize</a> | Sets buffer size     |
| Data Access Methods          |                      |
| <a href="#">GetData</a>      | Gets the data        |
| Data Update Methods          |                      |
| <a href="#">Refresh</a>      | Updates the data     |

## Create

Creates a timeseries with the specified parameters for access to the opening times of the bars in the history.

```
bool Create(  
    string      symbol,      // symbol  
    ENUM_TIMEFRAMES period    // period  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

### Returned value

true if successful, false if timeseries hasn't been created.

## BufferResize

Sets new size of the series.

```
virtual void BufferResize(  
    int size    // new size  
)
```

### Parameters

*size*

[in] New buffer size.

## GetData

Gets the element of timeseries by index.

```
datetime GetData(  
    int index      // index  
    ) const
```

### Parameters

*index*

[in] Index of the element needed.

### Returned value

The timeseries buffer element, or 0.

## GetData

Gets the element of timeseries by starting position and number of elements.

```
int GetData(  
    int start_pos,      // starting position  
    int count,          // number of elements to get  
    long& buffer        // target array  
    ) const
```

### Parameters

*start\_pos*

[in] Starting position of timeseries.

*count*

[in] Number of elements needed.

*buffer*

[in] Reference to the target array for the data.

### Returned value

>=0 if successful, -1 in the case of error.

## GetData

Gets the element of timeseries by starting time and number of elements.

```
int GetData(  
    datetime start_time, // starting time  
    int count,           // number of elements  
    long& buffer         // target array  
    ) const
```

### Parameters

*start\_time*

[in] Starting time.

*count*

[in] Number of elements needed.

*buffer*

[in] Reference to the target array for data.

#### Returned value

>=0 if successful, -1 in the case of error.

## GetData

Gets the element of timeseries by starting and stop times.

```
int  GetData(  
    datetime  start_time,    // starting time  
    datetime  stop_time,     // stop time  
    long&     buffer         // target array  
) const
```

#### Parameters

*start\_time*

[in] Starting time.

*stop\_time*

[in] Stop time.

*buffer*

[in] Reference to the target array for data

#### Returned value

>=0 if successful, -1 in the case of error.

## Refresh

Updates the data of timeseries.

```
virtual void Refresh(  
    int flags    // flags  
)
```

### Parameters

*flags*

[in] Timeframe flags.

## CiTickVolume

CiTickVolume is a class designed for access to tick volumes of the bars in the history.

### Description

The CiTickVolume class provides an access to tick volumes of the bars in the history.

### Declaration

```
class CiTickVolume: public CSeries
```

### Title

```
#include <Indicators\TimeSeries.mqh>
```

### Class Methods

| Create Methods               |                      |
|------------------------------|----------------------|
| <a href="#">Create</a>       | Creates a timeseries |
| <a href="#">BufferResize</a> | Sets buffer size     |
| Data Access Methods          |                      |
| <a href="#">GetData</a>      | Gets the data        |
| Data Update Methods          |                      |
| <a href="#">Refresh</a>      | Updates the data     |



## Create

Creates a timeseries with the specified parameters for access to the tick volumes of the bars in the history.

```
bool Create(  
    string      symbol,      // symbol  
    ENUM_TIMEFRAMES period    // period  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

### Returned value

true if successful, false if timeseries hasn't been created.

## BufferResize

Sets new size of the series.

```
virtual void BufferResize(  
    int size    // new size  
)
```

### Parameters

*size*

[in] New buffer size.

## GetData

Gets the element of timeseries by index.

```
datetime GetData(  
    int index      // index  
    ) const
```

### Parameters

*index*

[in] Index of the element needed.

### Returned value

The timeseries buffer element, or 0.

## GetData

Gets the element of timeseries by starting position and number of elements.

```
int GetData(  
    int start_pos,    // starting position  
    int count,        // number of elements to get  
    long& buffer      // target array  
    ) const
```

### Parameters

*start\_pos*

[in] Starting position of timeseries.

*count*

[in] Number of elements needed.

*buffer*

[in] Reference to the target array for the data.

### Returned value

>=0 if successful, -1 in the case of error.

## GetData

Gets the element of timeseries by starting time and number of elements.

```
int GetData(  
    datetime start_time,    // starting time  
    int count,              // number of elements  
    long& buffer            // target array  
    ) const
```

### Parameters

*start\_time*

[in] Starting time.

*count*

[in] Number of elements needed.

*buffer*

[in] Reference to the target array for data.

#### Returned value

>=0 if successful, -1 in the case of error.

## GetData

Gets the element of timeseries by starting and stop times.

```
int  GetData(  
    datetime  start_time,    // starting time  
    datetime  stop_time,     // stop time  
    long&     buffer         // target array  
) const
```

#### Parameters

*start\_time*

[in] Starting time.

*stop\_time*

[in] Stop time.

*buffer*

[in] Reference to the target array for data

#### Returned value

>=0 if successful, -1 in the case of error.

## Refresh

Updates the data of timeseries.

```
virtual void Refresh(  
    int flags    // flags  
)
```

### Parameters

*flags*

[in] Timeframe flags.

## CiRealVolume

CiRealVolume is a class designed for access to real volumes of the bars in the history.

### Description

The CiRealVolume class provides an access to real volumes of the bars in the history.

### Declaration

```
class CiRealVolume: public CSeries
```

### Title

```
#include <Indicators\TimeSeries.mqh>
```

### Class Methods

| Create Methods               |                      |
|------------------------------|----------------------|
| <a href="#">Create</a>       | Creates a timeseries |
| <a href="#">BufferResize</a> | Sets buffer size     |
| Data Access Methods          |                      |
| <a href="#">GetData</a>      | Gets the data        |
| Data Update Methods          |                      |
| <a href="#">Refresh</a>      | Updates the data     |

## Create

Creates a timeseries with the specified parameters for access to the real volumes of the bars in the history.

```
bool Create(  
    string      symbol,      // symbol  
    ENUM_TIMEFRAMES period    // period  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

### Returned value

true if successful, false if timeseries hasn't been created.

## BufferResize

Sets new size of the series.

```
virtual void BufferResize(  
    int size    // new size  
)
```

### Parameters

*size*

[in] New buffer size.



## GetData

Gets the element of timeseries by index.

```
datetime GetData(  
    int index      // index  
    ) const
```

### Parameters

*index*

[in] Index of the element needed.

### Returned value

The timeseries buffer element, or 0.

## GetData

Gets the element of timeseries by starting position and number of elements.

```
int GetData(  
    int start_pos,      // starting position  
    int count,          // number of elements to get  
    long& buffer        // target array  
    ) const
```

### Parameters

*start\_pos*

[in] Starting position of timeseries.

*count*

[in] Number of elements needed.

*buffer*

[in] Reference to the target array for the data.

### Returned value

>=0 if successful, -1 in the case of error.

## GetData

Gets the element of timeseries by starting time and number of elements.

```
int GetData(  
    datetime start_time, // starting time  
    int count,           // number of elements  
    long& buffer         // target array  
    ) const
```

### Parameters

*start\_time*

[in] Starting time.

*count*

[in] Number of elements needed.

*buffer*

[in] Reference to the target array for data.

#### Returned value

>=0 if successful, -1 in the case of error.

## GetData

Gets the element of timeseries by starting and stop times.

```
int  GetData(  
    datetime  start_time,    // starting time  
    datetime  stop_time,     // stop time  
    long&     buffer         // target array  
) const
```

#### Parameters

*start\_time*

[in] Starting time.

*stop\_time*

[in] Stop time.

*buffer*

[in] Reference to the target array for data

#### Returned value

>=0 if successful, -1 in the case of error.

## Refresh

Updates the data of timeseries.

```
virtual void Refresh(  
    int flags    // flags  
)
```

### Parameters

*flags*

[in] Timeframe flags.

## CiOpen

CiOpen is a class designed for access to open prices of the bars in the history.

### Description

The CiOpen class provides an access to open prices of the bars in the history.

### Declaration

```
class CiOpen: public CPriceSeries
```

### Title

```
#include <Indicators\TimeSeries.mqh>
```

### Class Methods

| Create Methods          |                      |
|-------------------------|----------------------|
| <a href="#">Create</a>  | Creates a timeseries |
| Data Access Methods     |                      |
| <a href="#">GetData</a> | Gets the data        |

## Create

Creates a timeseries with the specified parameters for access to the open prices of the bars in the history.

```
bool Create(  
    string      symbol,      // symbol  
    ENUM_TIMEFRAMES period    // period  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

### Returned value

true if successful, false if timeseries hasn't been created.

## GetData

Gets the element of timeseries by starting position and number of elements.

```
int  GetData(  
    int    start_pos,      // starting position  
    int    count,         // number of elements to get  
    double& buffer        // target array  
) const
```

### Parameters

*start\_pos*

[in] Starting position of timeseries.

*count*

[in] Number of elements needed.

*buffer*

[in] Reference to the target array for the data.

### Returned value

>=0 if successful, -1 in the case of error.

## GetData

Gets the element of timeseries by starting time and number of elements.

```
int  GetData(  
    datetime start_time,   // starting time  
    int      count,       // number of elements  
    double&  buffer       // target array  
) const
```

### Parameters

*start\_time*

[in] Starting time.

*count*

[in] Number of elements needed.

*buffer*

[in] Reference to the target array for data.

### Returned value

>=0 if successful, -1 in the case of error.

## GetData

Gets the element of timeseries by starting and stop times.

```
int  GetData(  
    datetime  start_time,      // starting time  
    datetime  stop_time,       // stop time  
    double&    buffer          // target array  
) const
```

### Parameters

*start\_time*

[in] Starting time.

*stop\_time*

[in] Stop time.

*buffer*

[in] Reference to the target array for data

### Returned value

>=0 if successful, -1 in the case of error.

## CiHigh

CiHigh is a class designed for access to high prices of the bars in the history.

### Description

The CiHigh class provides an access to high prices of the bars in the history.

### Declaration

```
class CiHigh: public CPriceSeries
```

### Title

```
#include <Indicators\TimeSeries.mqh>
```

### Class Methods

| Create Methods          |                      |
|-------------------------|----------------------|
| <a href="#">Create</a>  | Creates a timeseries |
| Data Access Methods     |                      |
| <a href="#">GetData</a> | Gets the data        |



## Create

Creates a timeseries with the specified parameters for access to the high prices of the bars in the history.

```
bool Create(  
    string      symbol,      // symbol  
    ENUM_TIMEFRAMES period    // period  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

### Returned value

true if successful, false if timeseries hasn't been created.

## GetData

Gets the element of timeseries by starting position and number of elements.

```
int  GetData(  
    int    start_pos,      // starting position  
    int    count,         // number of elements to get  
    double& buffer        // target array  
) const
```

### Parameters

*start\_pos*

[in] Starting position of timeseries.

*count*

[in] Number of elements needed.

*buffer*

[in] Reference to the target array for the data.

### Returned value

>=0 if successful, -1 in the case of error.

## GetData

Gets the element of timeseries by starting time and number of elements.

```
int  GetData(  
    datetime start_time,   // starting time  
    int      count,       // number of elements  
    double&  buffer       // target array  
) const
```

### Parameters

*start\_time*

[in] Starting time.

*count*

[in] Number of elements needed.

*buffer*

[in] Reference to the target array for data.

### Returned value

>=0 if successful, -1 in the case of error.

## GetData

Gets the element of timeseries by starting and stop times.

```
int  GetData(  
    datetime  start_time,      // starting time  
    datetime  stop_time,       // stop time  
    double&   buffer           // target array  
) const
```

### Parameters

*start\_time*

[in] Starting time.

*stop\_time*

[in] Stop time.

*buffer*

[in] Reference to the target array for data

### Returned value

>=0 if successful, -1 in the case of error.

## CiLow

CiLow is a class designed for access to low prices of the bars in the history.

### Description

The CiLow class provides an access to low prices of the bars in the history.

### Declaration

```
class CiLow: public CPriceSeries
```

### Title

```
#include <Indicators\TimeSeries.mqh>
```

### Class Methods

| Create Methods          |                      |
|-------------------------|----------------------|
| <a href="#">Create</a>  | Creates a timeseries |
| Data Access Methods     |                      |
| <a href="#">GetData</a> | Gets the data        |

## Create

Creates a timeseries with the specified parameters for access to the low prices of the bars in the history.

```
bool Create(  
    string      symbol,      // symbol  
    ENUM_TIMEFRAMES period    // period  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

### Returned value

true if successful, false if timeseries hasn't been created.

## GetData

Gets the element of timeseries by starting position and number of elements.

```
int  GetData(  
    int    start_pos,      // starting position  
    int    count,         // number of elements to get  
    double& buffer        // target array  
) const
```

### Parameters

*start\_pos*

[in] Starting position of timeseries.

*count*

[in] Number of elements needed.

*buffer*

[in] Reference to the target array for the data.

### Returned value

>=0 if successful, -1 in the case of error.

## GetData

Gets the element of timeseries by starting time and number of elements.

```
int  GetData(  
    datetime start_time,   // starting time  
    int      count,        // number of elements  
    double&  buffer        // target array  
) const
```

### Parameters

*start\_time*

[in] Starting time.

*count*

[in] Number of elements needed.

*buffer*

[in] Reference to the target array for data.

### Returned value

>=0 if successful, -1 in the case of error.

## GetData

Gets the element of timeseries by starting and stop times.

```
int  GetData(  
    datetime  start_time,      // starting time  
    datetime  stop_time,      // stop time  
    double&    buffer          // target array  
) const
```

### Parameters

*start\_time*

[in] Starting time.

*stop\_time*

[in] Stop time.

*buffer*

[in] Reference to the target array for data

### Returned value

>=0 if successful, -1 in the case of error.

## CiClose

CiClose is a class designed for access to close prices of the bars in the history.

### Description

The CiClose class provides an access to close prices of the bars in the history.

### Declaration

```
class CiClose: public CPriceSeries
```

### Title

```
#include <Indicators\TimeSeries.mqh>
```

### Class Methods

| Create Methods          |                      |
|-------------------------|----------------------|
| <a href="#">Create</a>  | Creates a timeseries |
| Data Access Methods     |                      |
| <a href="#">GetData</a> | Gets the data        |



## Create

Creates a timeseries with the specified parameters for access to the closing prices of the bars in the history.

```
bool Create(  
    string      symbol,      // symbol  
    ENUM_TIMEFRAMES period    // period  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

### Returned value

true if successful, false if timeseries hasn't been created.

## GetData

Gets the element of timeseries by starting position and number of elements.

```
int  GetData(  
    int    start_pos,      // starting position  
    int    count,         // number of elements to get  
    double& buffer        // target array  
) const
```

### Parameters

*start\_pos*

[in] Starting position of timeseries.

*count*

[in] Number of elements needed.

*buffer*

[in] Reference to the target array for the data.

### Returned value

>=0 if successful, -1 in the case of error.

## GetData

Gets the element of timeseries by starting time and number of elements.

```
int  GetData(  
    datetime start_time,   // starting time  
    int      count,        // number of elements  
    double&  buffer        // target array  
) const
```

### Parameters

*start\_time*

[in] Starting time.

*count*

[in] Number of elements needed.

*buffer*

[in] Reference to the target array for data.

### Returned value

>=0 if successful, -1 in the case of error.

## GetData

Gets the element of timeseries by starting and stop times.

```
int  GetData(  
    datetime  start_time,      // starting time  
    datetime  stop_time,       // stop time  
    double&    buffer          // target array  
) const
```

### Parameters

*start\_time*

[in] Starting time.

*stop\_time*

[in] Stop time.

*buffer*

[in] Reference to the target array for data

### Returned value

>=0 if successful, -1 in the case of error.

## Trend Indicator Classes

This group of chapters contains technical details of Trend Indicator classes of the MQL5 Standard Library and descriptions of all its key components.

| Class/group                        | Description                                |
|------------------------------------|--------------------------------------------|
| <a href="#"><u>CiADX</u></a>       | Average Directional Index                  |
| <a href="#"><u>CiADXWilder</u></a> | Average Directional Index by Welles Wilder |
| <a href="#"><u>CiBands</u></a>     | Bollinger Bands®                           |
| <a href="#"><u>CiEnvelopes</u></a> | Envelopes                                  |
| <a href="#"><u>CiIchimoku</u></a>  | Ichimoku Kinko Hyo                         |
| <a href="#"><u>CiMA</u></a>        | Moving Average                             |
| <a href="#"><u>CiSAR</u></a>       | Parabolic Stop And Reverse System          |
| <a href="#"><u>CiStdDev</u></a>    | Standard Deviation                         |
| <a href="#"><u>CiDEMA</u></a>      | Double Exponential Moving Average          |
| <a href="#"><u>CiTEMA</u></a>      | Triple Exponential Moving Average          |
| <a href="#"><u>CiFrAMA</u></a>     | Fractal Adaptive Moving Average            |
| <a href="#"><u>CiAMA</u></a>       | Adaptive Moving Average                    |
| <a href="#"><u>CiVIDyA</u></a>     | Variable Index DYnamic Average             |

## CiADX

CiADX is a class intended for using the Average Directional Index technical indicator.

### Description

The CiADX class provides the creation and access to the data of the Average Directional Index indicator.

### Declaration

```
class CiADX: public CIndicator
```

### Title

```
#include <Indicators\Trend.mqh>
```

### Class Methods

| Attributes                   |                                             |
|------------------------------|---------------------------------------------|
| <a href="#">MaPeriod</a>     | Returns the averaging period                |
| Create Methods               |                                             |
| <a href="#">Create</a>       | Creates the indicator                       |
| Data Access Methods          |                                             |
| <a href="#">Main</a>         | Returns the buffer element of the main line |
| <a href="#">Plus</a>         | Returns the buffer element of the +DI line  |
| <a href="#">Minus</a>        | Returns the buffer element of the -DI line  |
| Input/output                 |                                             |
| virtual <a href="#">Type</a> | Returns the object type identifier          |

## MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

### Returned value

Returns the averaging period, defined at the indicator creation.

## Create

It creates the indicator with specified parameters. Use [Refresh\(\)](#) and [GetData\(\)](#) to update and get the indicator values.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period        // Averaging period  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

*ma\_period*

[in] Averaging period.

### Returned value

true if successful, false if indicator hasn't been created.

## Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

Buffer element of the specified index if successful, or [EMPTY\\_VALUE](#) if there isn't any correct data.



## Plus

Returns the buffer element of the +DI line by the specified index.

```
double Plus(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

The buffer element of the +DI line of the specified index, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Minus

Returns the buffer element of the -DI line by the specified index.

```
double Minus (  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

The buffer element of the -DI line of the specified index, or `EMPTY_VALUE` if there isn't any correct data.

## Type

Returns the object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier ([IND\\_ADX](#) for CiADX).

## CiADXWilder

CiADXWilder is a class intended for using the Average Directional Index by Welles Wilder technical indicator.

### Description

The CiADXWilder class provides the creation and access to the data of the Average Directional Index by Welles Wilder indicator.

### Declaration

```
class CiADXWilder: public CIndicator
```

### Title

```
#include <Indicators\Trend.mqh>
```

### Class Methods

| Attributes                   |                                             |
|------------------------------|---------------------------------------------|
| <a href="#">MaPeriod</a>     | Returns the averaging period                |
| Create Methods               |                                             |
| <a href="#">Create</a>       | Creates the indicator                       |
| Data Access Methods          |                                             |
| <a href="#">Main</a>         | Returns the buffer element of the main line |
| <a href="#">Plus</a>         | Returns the buffer element of the +DI line  |
| <a href="#">Minus</a>        | Returns the buffer element of the -DI line  |
| Input/output                 |                                             |
| virtual <a href="#">Type</a> | Returns the object type identifier          |

## MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

### Returned value

Returns the averaging period, defined at the indicator creation.

## Create

It creates the indicator with specified parameters. Use [Refresh\(\)](#) and [GetData\(\)](#) to update and get the indicator values.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period        // Averaging period  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

*ma\_period*

[in] Averaging period.

### Returned value

true if successful, false if indicator hasn't been created.

## Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

Buffer element of the specified index if successful, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Plus

Returns the buffer element of the +DI line by the specified index.

```
double Plus(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

The buffer element of the +DI line of the specified index, or [EMPTY\\_VALUE](#) if there isn't any correct data.



## Minus

Returns the buffer element of the -DI line by the specified index.

```
double Minus (  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

The buffer element of the -DI line of the specified index, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Type

Returns the object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier ([IND\\_ADXW](#) for CiADXWilder).

## CiBands

CiBands is a class intended for using the Bollinger Bands® technical indicator.

### Description

The CiBands class provides the creation and access to the data of the Bollinger Bands indicator.

### Declaration

```
class CiBands: public CIndicator
```

### Title

```
#include <Indicators\Trend.mqh>
```

### Class Methods

| Attributes                   |                                              |
|------------------------------|----------------------------------------------|
| <a href="#">MaPeriod</a>     | Returns the averaging period                 |
| <a href="#">MaShift</a>      | Returns the horizontal shift                 |
| <a href="#">Deviation</a>    | Returns the deviation                        |
| <a href="#">Applied</a>      | Returns the price type or handle to apply    |
| <b>Create Methods</b>        |                                              |
| <a href="#">Create</a>       | Creates the indicator                        |
| <b>Data Access Methods</b>   |                                              |
| <a href="#">Main</a>         | Returns the buffer element of the base line  |
| <a href="#">Upper</a>        | Returns the buffer element of the upper line |
| <a href="#">Lower</a>        | Returns the buffer element of the lower line |
| <b>Input/output</b>          |                                              |
| virtual <a href="#">Type</a> | Returns the object type identifier           |

## MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

### Returned value

Returns the averaging period, defined at the indicator creation.

## MaShift

Returns the horizontal shift of the indicator.

```
int MaShift() const
```

### Returned value

Returns the horizontal shift value, defined at the indicator creation.

## Deviation

Returns the deviation.

```
double Deviation() const
```

### Returned value

Returns the deviation, defined at the indicator creation.

## Applied

Returns the price type or handle to apply.

```
int Applied() const
```

### Returned value

Price type or handle to apply, defined at the indicator creation.

## Create

It creates the indicator with specified parameters. Use [Refresh\(\)](#) and [GetData\(\)](#) to update and get the indicator values.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period,       // Averaging period  
    int             ma_shift,        // Shift  
    double          deviation,       // Deviation  
    int             applied           // applied price, or handle  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

*ma\_period*

[in] Averaging period.

*ma\_shift*

[in] Horizontal shift of the indicator.

*deviation*

[in] Deviation.

*applied*

[in] Volume type to apply.

### Returned value

true if successful, false if indicator hasn't been created.



## Base

Returns the buffer element of the base line by the specified index.

```
double Base(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

The buffer element of the Base line of the specified index, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Upper

Returns the buffer element of the upper line by the specified index.

```
double Upper (  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

The buffer element of the upper line of the specified index, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Lower

Returns the buffer element of the lower line by the specified index.

```
double Lower (  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

The buffer element of the lower line of the specified index, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Type

Returns the object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier ([IND\\_BANDS](#) for CiBands).

## CiEnvelopes

CiEnvelopes is a class intended for using the Envelopes technical indicator.

### Description

The CiEnvelopes class provides the creation and access to the data of the Envelopes indicator.

### Declaration

```
class CiEnvelopes: public CIndicator
```

### Title

```
#include <Indicators\Trend.mqh>
```

### Class Methods

| Attributes                   |                                              |
|------------------------------|----------------------------------------------|
| <a href="#">MaPeriod</a>     | Returns the averaging period                 |
| <a href="#">MaShift</a>      | Returns the horizontal shift                 |
| <a href="#">MaMethod</a>     | Returns the averaging method                 |
| <a href="#">Deviation</a>    | Returns the deviation                        |
| <a href="#">Applied</a>      | Returns the price type or handle to apply    |
| <b>Create Methods</b>        |                                              |
| <a href="#">Create</a>       | Creates the indicator                        |
| <b>Data Access Methods</b>   |                                              |
| <a href="#">Upper</a>        | Returns the buffer element of the upper line |
| <a href="#">Lower</a>        | Returns the buffer element of the lower line |
| <b>Input/output</b>          |                                              |
| virtual <a href="#">Type</a> | Returns the object type identifier           |

## MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

### Returned value

Returns the averaging period, defined at the indicator creation.

## MaShift

Returns the horizontal shift of the indicator.

```
int MaShift() const
```

### Returned value

Returns the horizontal shift value, defined at the indicator creation.

## MaMethod

Returns the averaging method.

```
ENUM_MA_METHOD MaMethod() const
```

### Returned value

Returns the averaging method, defined at the indicator creation.



## Deviation

Returns the value of deviation.

```
double Deviation() const
```

### Returned value

Returns the value of deviation, defined at the indicator creation.

## Applied

Returns the price type or handle to apply.

```
int Applied() const
```

### Returned value

Price type or handle to apply, defined at the indicator creation.

## Create

It creates the indicator with specified parameters. Use [Refresh\(\)](#) and [GetData\(\)](#) to update and get the indicator values.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period,        // Averaging period  
    int             ma_shift,         // Horizontal shift  
    ENUM_MA_METHOD  ma_method,        // Averaging method  
    int             applied,          // Price type or handle to apply  
    double          deviation         // Deviation  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

*ma\_period*

[in] Averaging period.

*ma\_shift*

[in] Horizontal shift.

*ma\_method*

[in] Averaging method ([ENUM\\_MA\\_METHOD](#) enumeration).

*applied*

[in] Price type of handle to apply.

*deviation*

[in] Deviation.

### Returned value

true if successful, false if indicator hasn't been created.

## Upper

Returns the buffer element of the upper line by the specified index.

```
double Upper (  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

The buffer element of the upper line of the specified index, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Lower

Returns the buffer element of the lower line by the specified index.

```
double Lower (  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

The buffer element of the lower line of the specified index, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Type

Returns the object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier ([IND\\_ENVELOPES](#) for CiEnvelopes).

## Cilchimoku

Cilchimoku is a class intended for using the Ichimoku Kinko Hyo technical indicator.

### Description

The Cilchimoku class provides the creation, setup and access to the data of the Ichimoku Kinko Hyo indicator.

### Declaration

```
class CiIchimoku: public CIndicator
```

### Title

```
#include <Indicators\Trend.mqh>
```

### Class Methods

| Attributes                        |                                                    |
|-----------------------------------|----------------------------------------------------|
| <a href="#">TenkanSenPeriod</a>   | Returns the TenkanSen period                       |
| <a href="#">KijunSenPeriod</a>    | Returns the KijunSen period                        |
| <a href="#">SenkouSpanBPeriod</a> | Returns the SenkouSpanB period                     |
| <b>Create Methods</b>             |                                                    |
| <a href="#">Create</a>            | Creates the indicator                              |
| <b>Data Access Methods</b>        |                                                    |
| <a href="#">TenkanSen</a>         | Returns the buffer element of the TenkanSen line   |
| <a href="#">KijunSen</a>          | Returns the buffer element of the KijunSen line    |
| <a href="#">SenkouSpanA</a>       | Returns the buffer element of the SenkouSpanA line |
| <a href="#">SenkouSpanB</a>       | Returns the buffer element of the SenkouSpanB line |
| <a href="#">ChinkouSpan</a>       | Returns the buffer element of the ChinkouSpan line |
| <b>Input/output</b>               |                                                    |
| virtual <a href="#">Type</a>      | Returns the object type identifier                 |

## TenkanSenPeriod

Returns the TenkanSen period.

```
int TenkanSenPeriod() const
```

### Returned value

Returns the TenkanSen period, defined at the indicator creation.



## KijunSenPeriod

Returns the KijunSen period.

```
int KijunSenPeriod() const
```

### Returned value

Returns the KijunSen period, defined at the indicator creation.

## SenkouSpanBPeriod

Returns the SenkouSpanB period.

```
int SenkouSpanBPeriod() const
```

### Returned value

Returns the SenkouSpanB period, defined at the indicator creation.

## Create

It creates the indicator with specified parameters. Use [Refresh\(\)](#) and [GetData\(\)](#) to update and get the indicator values.

```
bool Create(  
    string          symbol,           // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             tenkan_sen,       // Period of TenkanSen  
    int             kijun_sen,        // Period of KijunSen  
    int             senkou_span_b     // Period of SenkouSpanB  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

*tenkan\_sen*

[in] Period of TenkanSen.

*kijun\_sen*

[in] Period of KijunSen.

*senkou\_span\_b*

[in] Period of SenkouSpanB.

### Returned value

true if successful, false if indicator hasn't been created.

## TenkanSen

Returns the buffer element of the TenkanSen line by the specified index.

```
double TenkanSen(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

The buffer element of the TenkanSen line of the specified index, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## KijunSen

Returns the buffer element of the KijunSen line by the specified index.

```
double KijunSen(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

The buffer element of the KijunSen line of the specified index, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## SenkouSpanA

Returns the buffer element of the SenkouSpanA line by the specified index.

```
double SenkouSpanA(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

The buffer element of the SenkouSpanA line of the specified index, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## SenkouSpanB

Returns the buffer element of the SenkouSpanB line by the specified index.

```
double SenkouSpanB(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

The buffer element of the SenkouSpanB line of the specified index, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## ChinkouSpan

Returns the buffer element of the ChinkouSpan line by the specified index.

```
double ChinkouSpan(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

The buffer element of the ChinkouSpan line of the specified index, or [EMPTY\\_VALUE](#) if there isn't any correct data.



## Type

Returns the object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier ([IND\\_ICHIMOKU](#) for Cilchimoku).

## CiMA

CiMA is a class intended for using the Moving Average technical indicator.

### Description

The CiMA class provides the creation, setup and access to the data of the Moving Average indicator.

### Declaration

```
class CiMA: public CIndicator
```

### Title

```
#include <Indicators\Trend.mqh>
```

### Class Methods

| Attributes                   |                                           |
|------------------------------|-------------------------------------------|
| <a href="#">MaPeriod</a>     | Returns the averaging period              |
| <a href="#">MaShift</a>      | Returns the horizontal shift              |
| <a href="#">MaMethod</a>     | Returns the averaging method              |
| <a href="#">Applied</a>      | Returns the price type or handle to apply |
| <b>Create Methods</b>        |                                           |
| <a href="#">Create</a>       | Creates the indicator                     |
| <b>Data Access Methods</b>   |                                           |
| <a href="#">Main</a>         | Returns the buffer element                |
| <b>Input/output</b>          |                                           |
| virtual <a href="#">Type</a> | Returns the object type identifier        |

## MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

### Returned value

Returns the averaging period, defined at the indicator creation.

## MaShift

Returns the horizontal shift of the indicator.

```
int MaShift() const
```

### Returned value

Returns the horizontal shift value, defined at the indicator creation.

## MaMethod

Returns the averaging method.

```
ENUM_MA_METHOD MaMethod() const
```

### Returned value

Returns the averaging method (value of [ENUM\\_MA\\_METHOD](#) enumeration), defined at the indicator creation.

## Applied

Returns the price type or handle to apply.

```
int Applied() const
```

### Returned value

Price type or handle to apply, defined at the indicator creation.

## Create

It creates the indicator with specified parameters. Use [Refresh\(\)](#) and [GetData\(\)](#) to update and get the indicator values.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period,       // Averaging period  
    int             ma_shift,        // Horizontal shift  
    ENUM_MA_METHOD  ma_method,       // Averaging method  
    int             applied           // Price type of handle to apply  
)
```

### Parameters

*string*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

*ma\_period*

[in] Averaging period.

*ma\_shift*

[in] Horizontal shift.

*ma\_method*

[in] Averaging method ([ENUM\\_MA\\_METHOD](#) enumeration).

*applied*

[in] Price type or handle to apply.

### Returned value

true if successful, false if indicator hasn't been created.

## Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

Buffer element of the specified index if successful, or [EMPTY\\_VALUE](#) if there isn't any correct data.



## Type

Returns the object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier ([IND\\_MA](#) for CiMA).

## CiSAR

CiSAR is a class intended for using the Parabolic Stop And Reverse System technical indicator.

### Description

The CiSAR class provides the creation, setup and access to the data of the Parabolic Stop And Reverse System indicator.

### Declaration

```
class CiSAR: public CIndicator
```

### Title

```
#include <Indicators\Trend.mqh>
```

### Class Methods

| Attributes                   |                                              |
|------------------------------|----------------------------------------------|
| <a href="#">SarStep</a>      | Returns the step for the velocity increasing |
| <a href="#">Maximum</a>      | Returns the coefficient of price following   |
| Create Methods               |                                              |
| <a href="#">Create</a>       | Creates the indicator                        |
| Data Access Methods          |                                              |
| <a href="#">Main</a>         | Returns the buffer element                   |
| Input/output                 |                                              |
| virtual <a href="#">Type</a> | Returns the object type identifier           |

## SarStep

Returns the step for the velocity increasing (acceleration coefficient).

```
double SarStep() const
```

### Returned value

The step for the velocity increasing, defined at the indicator creation.

## Maximum

Returns the coefficient of price following.

```
double Maximum() const
```

### Returned value

The price following coefficient, defined at the indicator creation.

## Create

It creates the indicator with specified parameters. Use [Refresh\(\)](#) and [GetData\(\)](#) to update and get the indicator values.

```
bool Create(  
    string          symbol,      // Symbol  
    ENUM_TIMEFRAMES period,     // Period  
    double          step,       // Step  
    double          maximum     // Coefficient  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

*step*

[in] Step for the velocity increasing.

*maximum*

[in] Price following coefficient.

### Returned value

true if successful, false if indicator hasn't been created.

## Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

Buffer element of the specified index if successful, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Type

Returns the object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier ([IND\\_SAR](#) for CiSAR).

## CiStdDev

CiStdDev is a class intended for using the Standard Deviation technical indicator.

### Description

The CiStdDev class provides the creation, setup and access to the data of the Standard Deviation indicator.

### Declaration

```
class CiStdDev: public CIndicator
```

### Title

```
#include <Indicators\Trend.mqh>
```

### Class Methods

| Attributes                   |                                           |
|------------------------------|-------------------------------------------|
| <a href="#">MaPeriod</a>     | Returns the averaging period              |
| <a href="#">MaShift</a>      | Returns the horizontal shift              |
| <a href="#">MaMethod</a>     | Returns the averaging method              |
| <a href="#">Applied</a>      | Returns the price type or handle to apply |
| Create Methods               |                                           |
| <a href="#">Create</a>       | Creates the indicator                     |
| Data Access Methods          |                                           |
| <a href="#">Main</a>         | Returns the buffer element                |
| Input/output                 |                                           |
| virtual <a href="#">Type</a> | Returns the object type identifier        |



## MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

### Returned value

Returns the averaging period, defined at the indicator creation.

## MaShift

Returns the horizontal shift of the indicator.

```
int MaShift() const
```

### Returned value

Returns the horizontal shift value, defined at the indicator creation.

## MaMethod

Returns the averaging method.

```
ENUM_MA_METHOD MaMethod() const
```

### Returned value

Returns the averaging method (value of [ENUM\\_MA\\_METHOD](#) enumeration), defined at the indicator creation.

## Applied

Returns the price type or handle to apply.

```
int Applied() const
```

### Returned value

Price type or handle to apply, defined at the indicator creation.

## Create

It creates the indicator with specified parameters. Use [Refresh\(\)](#) and [GetData\(\)](#) to update and get the indicator values.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period,       // Averaging period  
    int             ma_shift,        // Horizontal shift  
    ENUM_MA_METHOD  ma_method,       // Averaging method  
    int             applied           // Price type or handle to apply  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

*ma\_period*

[in] Averaging period.

*ma\_shift*

[in] Horizontal shift.

*ma\_method*

[in] Averaging method ([ENUM\\_MA\\_METHOD](#) enumeration).

*applied*

[in] Price type or handle to apply.

### Returned value

true if successful, false if indicator hasn't been created.

## Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

Buffer element of the specified index if successful, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Type

Returns the object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier ([IND\\_STDDEV](#) for CStdDev).

## CiDEMA

CiDEMA is a class intended for using the Double Exponential Moving Average technical indicator.

### Description

The CiDEMA class provides the creation, setup and access to the data of the Double Exponential Moving Average indicator.

### Declaration

```
class CiDEMA: public CIndicator
```

### Title

```
#include <Indicators\Trend.mqh>
```

### Class Methods

| Attributes                   |                                           |
|------------------------------|-------------------------------------------|
| <a href="#">MaPeriod</a>     | Returns the averaging period              |
| <a href="#">IndShift</a>     | Returns the horizontal shift              |
| <a href="#">Applied</a>      | Returns the price type or handle to apply |
| Create Methods               |                                           |
| <a href="#">Create</a>       | Creates the indicator                     |
| Data Access Methods          |                                           |
| <a href="#">Main</a>         | Returns the buffer element                |
| Input/output                 |                                           |
| virtual <a href="#">Type</a> | Returns the object type identifier        |



## MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

### Returned value

Returns the averaging period, defined at the indicator creation.

## IndShift

Returns the horizontal shift of the indicator.

```
int IndShift () const
```

### Returned value

Returns the horizontal shift value, defined at the indicator creation.

## Applied

Returns the price type or handle to apply.

```
int Applied() const
```

### Returned value

Price type or handle to apply, defined at the indicator creation.

## Create

It creates the indicator with specified parameters. Use [Refresh\(\)](#) and [GetData\(\)](#) to update and get the indicator values.

```
bool Create(  
    string          string,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period,        // Averaging period  
    int             ind_shift,        // Shift  
    int             applied           // Price type of handle to apply  
)
```

### Parameters

*string*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

*ma\_period*

[in] Averaging period.

*ind\_shift*

[in] Horizontal shift.

*applied*

[in] Price type or handle to apply.

### Returned value

true if successful, false if indicator hasn't been created.

## Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

Buffer element of the specified index if successful, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Type

Returns the object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier ([IND\\_DEMA](#) for CiDEMA).

## CiTEMA

CiTEMA is a class intended for using the Triple Exponential Moving Average technical indicator.

### Description

The CiTEMA class provides the creation, setup and access to the data of the Triple Exponential Moving Average indicator.

### Declaration

```
class CiTEMA: public CIndicator
```

### Title

```
#include <Indicators\Trend.mqh>
```

### Class Methods

| Attributes                   |                                           |
|------------------------------|-------------------------------------------|
| <a href="#">MaPeriod</a>     | Returns the averaging period              |
| <a href="#">IndShift</a>     | Returns the horizontal shift              |
| <a href="#">Applied</a>      | Returns the price type or handle to apply |
| <b>Create Methods</b>        |                                           |
| <a href="#">Create</a>       | Creates the indicator                     |
| <b>Data Access Methods</b>   |                                           |
| <a href="#">Main</a>         | Returns the buffer element                |
| <b>Input/output</b>          |                                           |
| virtual <a href="#">Type</a> | Returns the object type identifier        |

## MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

### Returned value

Returns the averaging period, defined at the indicator creation.



## IndShift

Returns the horizontal shift of the indicator.

```
int IndShift () const
```

### Returned value

Returns the horizontal shift value, defined at the indicator creation.

## Applied

Returns the price type or handle to apply.

```
int Applied() const
```

### Returned value

Price type or handle to apply, defined at the indicator creation.

## Create

It creates the indicator with specified parameters. Use [Refresh\(\)](#) and [GetData\(\)](#) to update and get the indicator values.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period,        // Averaging period  
    int             ma_shift,         // Offset  
    int             applied           // Price type of handle to apply  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

*ma\_period*

[in] Averaging period.

*ma\_shift*

[in] Horizontal shift.

*applied*

[in] Price type or handle to apply.

### Returned value

true if successful, false if indicator hasn't been created.

## Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

Buffer element of the specified index if successful, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Type

Returns the object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier ([IND\\_TEMA](#) for CiTEMA).

## CiFrAMA

CiFrAMA is a class intended for using the Fractal Adaptive Moving Average technical indicator.

### Description

The CiFrAMA class provides the creation, setup and access to the data of the Fractal Adaptive Moving Average indicator.

### Declaration

```
class CiFrAMA: public CIndicator
```

### Title

```
#include <Indicators\Trend.mqh>
```

### Class Methods

| Attributes                   |                                           |
|------------------------------|-------------------------------------------|
| <a href="#">MaPeriod</a>     | Returns the averaging period              |
| <a href="#">IndShift</a>     | Returns the horizontal shift              |
| <a href="#">Applied</a>      | Returns the price type or handle to apply |
| <b>Create Methods</b>        |                                           |
| <a href="#">Create</a>       | Creates the indicator                     |
| <b>Data Access Methods</b>   |                                           |
| <a href="#">Main</a>         | Returns the buffer element                |
| <b>Input/output</b>          |                                           |
| virtual <a href="#">Type</a> | Returns the object type identifier        |

## MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

### Returned value

Returns the averaging period, defined at the indicator creation.

## IndShift

Returns the horizontal shift of the indicator.

```
int IndShift () const
```

### Returned value

Returns the horizontal shift value, defined at the indicator creation.



## Applied

Returns the price type or handle to apply.

```
int Applied() const
```

### Returned value

Price type or handle to apply, defined at the indicator creation.

## Create

It creates the indicator with specified parameters. Use [Refresh\(\)](#) and [GetData\(\)](#) to update and get the indicator values.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period,        // Averaging period  
    int             ma_shift,         // Offset  
    int             applied           // Price type of handle to apply  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

*ma\_period*

[in] Averaging period.

*ma\_shift*

[in] Horizontal shift.

*applied*

[in] Price type or handle to apply.

### Returned value

true if successful, false if indicator hasn't been created.

## Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

Buffer element of the specified index if successful, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Type

Returns the object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier ([IND\\_FRAMA](#) for CiFrAMA).

## CiAMA

CiAMA is a class intended for using the Adaptive Moving Average technical indicator.

### Description

The CiAMA class provides the creation, setup and access to the data of the Adaptive Moving Average indicator.

### Declaration

```
class CiAMA: public CIndicator
```

### Title

```
#include <Indicators\Trend.mqh>
```

### Class Methods

| Attributes                    |                                               |
|-------------------------------|-----------------------------------------------|
| <a href="#">MaPeriod</a>      | Returns the averaging period                  |
| <a href="#">FastEmaPeriod</a> | Returns the averaging period for the fast EMA |
| <a href="#">SlowEmaPeriod</a> | Returns the averaging period for the slow EMA |
| <a href="#">IndShift</a>      | Returns the horizontal shift                  |
| <a href="#">Applied</a>       | Returns the price type or handle to apply     |
| <b>Create Methods</b>         |                                               |
| <a href="#">Create</a>        | Creates the indicator                         |
| <b>Data Access Methods</b>    |                                               |
| <a href="#">Main</a>          | Returns the buffer element                    |
| <b>Input/output</b>           |                                               |
| virtual <a href="#">Type</a>  | Returns the object type identifier            |

## MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

### Returned value

Returns the averaging period, defined at the indicator creation.

## FastEmaPeriod

Returns the averaging period for the fast EMA.

```
int FastEmaPeriod() const
```

### Returned value

Returns the averaging period for the fast EMA, defined at the indicator creation.

## SlowEmaPeriod

Returns the averaging period for the slow EMA.

```
int SlowEmaPeriod() const
```

### Returned value

Returns the averaging period for the slow EMA, defined at the indicator creation.



## IndShift

Returns the horizontal shift of the indicator.

```
int IndShift () const
```

### Returned value

Returns the horizontal shift value, defined at the indicator creation.

## Applied

Returns the price type or handle to apply.

```
int Applied() const
```

### Returned value

Price type or handle to apply, defined at the indicator creation.

## Create

It creates the indicator with specified parameters. Use [Refresh\(\)](#) and [GetData\(\)](#) to update and get the indicator values.

```
bool Create(  
    string          string,           // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period,        // Averaging period  
    int             fast_ema_period,  // Fast EMA period  
    int             slow_ema_period,  // Slow EMA period  
    int             ind_shift,        // Shift  
    int             applied           // Price type or handle to apply  
)
```

### Parameters

*string*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

*ma\_period*

[in] Averaging period.

*fast\_ema\_period*

[in] Fast EMA averaging period.

*slow\_ema\_period*

[in] Slow EMA averaging period.

*ind\_shift*

[in] Horizontal shift.

*applied*

[in] Price type or handle to apply.

### Returned value

true if successful, false if indicator hasn't been created.

## Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

Buffer element of the specified index if successful, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Type

Returns the object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier ([IND\\_AMA](#) for CiAMA).

## CiVIDyA

CiVIDyA is a class intended for using the Variable Index DYnamic Average technical indicator.

### Description

The CiVIDyA class provides the creation, setup and access to the data of the Variable Index DYnamic Average indicator.

### Declaration

```
class CiVIDyA: public CIndicator
```

### Title

```
#include <Indicators\Trend.mqh>
```

### Class Methods

| Attributes                   |                                           |
|------------------------------|-------------------------------------------|
| <a href="#">CmoPeriod</a>    | Returns the period for Momentum           |
| <a href="#">EmaPeriod</a>    | Returns the averaging period for EMA      |
| <a href="#">IndShift</a>     | Returns the horizontal shift              |
| <a href="#">Applied</a>      | Returns the price type or handle to apply |
| Create Methods               |                                           |
| <a href="#">Create</a>       | Creates the indicator                     |
| Data Access Methods          |                                           |
| <a href="#">Main</a>         | Returns the buffer element                |
| Input/output                 |                                           |
| virtual <a href="#">Type</a> | Returns the object type identifier        |

## CmoPeriod

Returns the period for Momentum.

```
int CmoPeriod() const
```

### Returned value

Returns the period for Momentum, defined at the indicator creation.

## EmaPeriod

Returns the averaging period for EMA.

```
int EmaPeriod() const
```

### Returned value

Returns the averaging period for EMA, defined at the indicator creation.



## IndShift

Returns the horizontal shift of the indicator.

```
int IndShift () const
```

### Returned value

Returns the horizontal shift value, defined at the indicator creation.

## Applied

Returns the price type or handle to apply.

```
int Applied() const
```

### Returned value

Price type or handle to apply, defined at the indicator creation.

## Create

It creates the indicator with specified parameters. Use [Refresh\(\)](#) and [GetData\(\)](#) to update and get the indicator values.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             cmo_period,      // Momentum period  
    int             ema_period,      // Averaging period  
    int             ind_shift,       // Shift  
    int             applied           // Price type or handle to apply  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

*cmo\_period*

[in] Momentum period.

*ema\_period*

[in] Averaging period.

*ind\_shift*

[in] Horizontal shift.

*applied*

[in] Price type or handle to apply.

### Returned value

true if successful, false if indicator hasn't been created.

## Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

Buffer element of the specified index if successful, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Type

Returns the object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier ([IND\\_VIDYA](#) for CiViDyA).

## Oscillators

This group of chapters contains the technical details of Oscillators classes of the MQL5 Standard Library and descriptions of all its key components.

| Class/group                         | Description                                   |
|-------------------------------------|-----------------------------------------------|
| <a href="#"><u>CiATR</u></a>        | Average True Range                            |
| <a href="#"><u>CiBearsPower</u></a> | Bears Power                                   |
| <a href="#"><u>CiBullsPower</u></a> | Bulls Power                                   |
| <a href="#"><u>CiCCI</u></a>        | Commodity Channel Index                       |
| <a href="#"><u>CiChaikin</u></a>    | Chaikin Oscillator                            |
| <a href="#"><u>CiDeMarker</u></a>   | DeMarker                                      |
| <a href="#"><u>CiForce</u></a>      | Force Index                                   |
| <a href="#"><u>CiMACD</u></a>       | Moving Averages Convergence-Divergence        |
| <a href="#"><u>CiMomentum</u></a>   | Momentum                                      |
| <a href="#"><u>CiOsMA</u></a>       | Moving Average of Oscillator (MACD histogram) |
| <a href="#"><u>CiRSI</u></a>        | Relative Strength Index                       |
| <a href="#"><u>CiRVI</u></a>        | Relative Vigor Index                          |
| <a href="#"><u>CiStochastic</u></a> | Stochastic Oscillator                         |
| <a href="#"><u>CiWPR</u></a>        | Williams' Percent Range                       |
| <a href="#"><u>CiTriX</u></a>       | Triple Exponential Moving Averages Oscillator |

## CiATR

CiATR is a class intended for using the Average True Range technical indicator.

### Description

The CiATR class provides the creation, setup and access to the data of the Average True Range indicator.

### Declaration

```
class CiATR: public CIndicator
```

### Title

```
#include <Indicators\Oscillators.mqh>
```

### Class Methods

| Attributes                   |                                    |
|------------------------------|------------------------------------|
| <a href="#">MaPeriod</a>     | Returns the averaging period       |
| Create Methods               |                                    |
| <a href="#">Create</a>       | Creates the indicator              |
| Data Access Methods          |                                    |
| <a href="#">Main</a>         | Returns the buffer element         |
| Input/output                 |                                    |
| virtual <a href="#">Type</a> | Returns the object type identifier |

## MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

### Returned value

Returns the averaging period, defined at the indicator creation.



## Create

It creates the indicator with specified parameters. Use [Refresh\(\)](#) and [GetData\(\)](#) to update and get the indicator values.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period        // Averaging period  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

*ma\_period*

[in] Averaging period.

### Returned value

true if successful, false if indicator hasn't been created.

## Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

Buffer element of the specified index if successful, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Type

Returns the object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier ([IND\\_ATR](#) for CiATR).

## CiBearsPower

CiBearsPower is a class intended for using the Bears Power technical indicator.

### Description

The CiBearsPower class provides the creation, setup and access to the data of the Bears Power indicator.

### Declaration

```
class CiBearsPower: public CIndicator
```

### Title

```
#include <Indicators\Oscillators.mqh>
```

### Class Methods

| Attributes                   |                                    |
|------------------------------|------------------------------------|
| <a href="#">MaPeriod</a>     | Returns the averaging period       |
| Create Methods               |                                    |
| <a href="#">Create</a>       | Creates the indicator              |
| Data Access Methods          |                                    |
| <a href="#">Main</a>         | Returns the buffer element         |
| Input/output                 |                                    |
| virtual <a href="#">Type</a> | Returns the object type identifier |

## MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

### Returned value

Returns the averaging period, defined at the indicator creation.

## Create

It creates the indicator with specified parameters. Use [Refresh\(\)](#) and [GetData\(\)](#) to update and get the indicator values.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period        // Averaging period  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

*ma\_period*

[in] Averaging period.

### Returned value

true if successful, false if indicator hasn't been created.

## Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

Buffer element of the specified index if successful, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Type

Returns the object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier ([IND\\_BEARS](#) for CiBearsPower).



## CiBullsPower

CiBullsPower is a class intended for using the Bulls Power technical indicator.

### Description

The CiBullsPower class provides the creation, setup and access to the data of the Bulls Power indicator.

### Declaration

```
class CiBullsPower: public CIndicator
```

### Title

```
#include <Indicators\Oscillators.mqh>
```

### Class Methods

| Attributes                   |                                    |
|------------------------------|------------------------------------|
| <a href="#">MaPeriod</a>     | Returns the averaging period       |
| Create Methods               |                                    |
| <a href="#">Create</a>       | Creates the indicator              |
| Data Access Methods          |                                    |
| <a href="#">Main</a>         | Returns the buffer element         |
| Input/output                 |                                    |
| virtual <a href="#">Type</a> | Returns the object type identifier |

## MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

### Returned value

Returns the averaging period, defined at the indicator creation.

## Create

It creates the indicator with specified parameters. Use [Refresh\(\)](#) and [GetData\(\)](#) to update and get the indicator values.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period        // Averaging period  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

*ma\_period*

[in] Averaging period.

### Returned value

true if successful, false if indicator hasn't been created.

## Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

Buffer element of the specified index if successful, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Type

Returns the object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier ([IND\\_BULLS](#) for CiBullsPower).

## CiCCI

CiCCI is a class intended for using the Commodity Channel Index technical indicator.

### Description

The CiCCI class provides the creation, setup and access to the data of the Commodity Channel Index indicator.

### Declaration

```
class CiCCI: public CIndicator
```

### Title

```
#include <Indicators\Oscillators.mqh>
```

### Class Methods

| Attributes                   |                                           |
|------------------------------|-------------------------------------------|
| <a href="#">MaPeriod</a>     | Returns the averaging period              |
| <a href="#">Applied</a>      | Returns the price type or handle to apply |
| Create Methods               |                                           |
| <a href="#">Create</a>       | Creates the indicator                     |
| Data Access Methods          |                                           |
| <a href="#">Main</a>         | Returns the buffer element                |
| Input/output                 |                                           |
| virtual <a href="#">Type</a> | Returns the object type identifier        |

## MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

### Returned value

Returns the averaging period, defined at the indicator creation.

## Applied

Returns the price type or handle to apply.

```
int Applied() const
```

### Returned value

Price type or handle to apply, defined at the indicator creation.



## Create

It creates the indicator with specified parameters. Use [Refresh\(\)](#) and [GetData\(\)](#) to update and get the indicator values.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period,        // Averaging period  
    int             applied           // Price type or handle to apply  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

*ma\_period*

[in] Averaging period.

*applied*

[in] Price type or handle to apply.

### Returned value

true if successful, false if indicator hasn't been created.

## Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

Buffer element of the specified index if successful, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Type

Returns the object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier ([IND\\_CCI](#) for CiCCI).

## CiChaikin

CiChaikin is a class intended for using the Chaikin Oscillator technical indicator.

### Description

The CiChaikin class provides the creation, setup and access to the data of the Chaikin Oscillator indicator.

### Declaration

```
class CiChaikin: public CIndicator
```

### Title

```
#include <Indicators\Oscillators.mqh>
```

### Class Methods

| Attributes                   |                                              |
|------------------------------|----------------------------------------------|
| <a href="#">FastMaPeriod</a> | Returns the averaging period for the fast MA |
| <a href="#">SlowMaPeriod</a> | Returns the averaging period for the slow MA |
| <a href="#">MaMethod</a>     | Returns the averaging method                 |
| <a href="#">Applied</a>      | Returns the price type or handle to apply    |
| Create Methods               |                                              |
| <a href="#">Create</a>       | Creates the indicator                        |
| Data Access Methods          |                                              |
| <a href="#">Main</a>         | Returns the buffer element                   |
| Input/output                 |                                              |
| virtual <a href="#">Type</a> | Returns the object type identifier           |

## FastMaPeriod

Returns the averaging period for the fast EMA.

```
int FastMaPeriod() const
```

### Returned value

Returns the averaging period for the fast EMA, defined at the indicator creation.

## SlowMaPeriod

Returns the averaging period for the slow EMA.

```
int SlowMaPeriod() const
```

### Returned value

Returns the averaging period for the slow EMA, defined at the indicator creation.

## MaMethod

Returns the averaging method.

```
ENUM_MA_METHOD MaMethod() const
```

### Returned value

Returns the averaging method, defined at the indicator creation.

## Applied

Returns the volume type to apply.

```
ENUM_APPLIED_VOLUME Applied() const
```

### Returned value

Volume type to apply, defined at the indicator creation.



## Create

It creates the indicator with specified parameters. Use [Refresh\(\)](#) and [GetData\(\)](#) to update and get the indicator values.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             fast_ma_period,  // Fast EMA period  
    int             slow_ma_period,  // Slow EMA period  
    ENUM_MA_METHOD  ma_method,       // Averaging method  
    ENUM_APPLIED_VOLUME applied      // Volume type to apply  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

*fast\_ma\_period*

[in] Period for fast EMA.

*slow\_ma\_period*

[in] Period for slow EMA.

*ma\_method*

[in] Averaging method ([ENUM\\_MA\\_METHOD](#) enumeration).

*applied*

[in] Volume type to apply ([ENUM\\_APPLIED\\_VOLUME](#) enumeration).

### Returned value

true if successful, false if indicator hasn't been created.

## Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

Buffer element of the specified index if successful, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Type

Returns the object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier ([IND\\_CHAIKIN](#) for CiChaikin).

## CiDeMarker

CiDeMarker is a class intended for using the DeMarker technical indicator.

### Description

The CiDeMarker class provides the creation, setup and access to the data of the DeMarker indicator.

### Declaration

```
class CiDeMarker: public CIndicator
```

### Title

```
#include <Indicators\Oscillators.mqh>
```

### Class Methods

| Attributes                   |                                    |
|------------------------------|------------------------------------|
| <a href="#">MaPeriod</a>     | Returns the averaging period       |
| Create Methods               |                                    |
| <a href="#">Create</a>       | Creates the indicator              |
| Data Access Methods          |                                    |
| <a href="#">Main</a>         | Returns the buffer element         |
| Input/output                 |                                    |
| virtual <a href="#">Type</a> | Returns the object type identifier |

## MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

### Returned value

Returns the averaging period, defined at the indicator creation.

## Create

It creates the indicator with specified parameters. Use [Refresh\(\)](#) and [GetData\(\)](#) to update and get the indicator values.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period        // Averaging period  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

*ma\_period*

[in] Averaging period.

### Returned value

true if successful, false if indicator hasn't been created.

## Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

Buffer element of the specified index if successful, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Type

Returns the object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier ([IND\\_DEMARKER](#) for CiDeMarker).



## CiForce

CiForce is a class intended for using the Force Index technical indicator.

### Description

The CiForce class provides the creation, setup and access to the data of the Force Index indicator.

### Declaration

```
class CiForce: public CIndicator
```

### Title

```
#include <Indicators\Oscillators.mqh>
```

### Class Methods

| Attributes                   |                                           |
|------------------------------|-------------------------------------------|
| <a href="#">MaPeriod</a>     | Returns the averaging period              |
| <a href="#">MaMethod</a>     | Returns the averaging method              |
| <a href="#">Applied</a>      | Returns the price type or handle to apply |
| Create Methods               |                                           |
| <a href="#">Create</a>       | Creates the indicator                     |
| Data Access Methods          |                                           |
| <a href="#">Main</a>         | Returns the buffer element                |
| Input/output                 |                                           |
| virtual <a href="#">Type</a> | Returns the object type identifier        |

## MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

### Returned value

Returns the averaging period, defined at the indicator creation.

## MaMethod

Returns the averaging method.

```
ENUM_MA_METHOD MaMethod() const
```

### Returned value

Returns the averaging method, defined at the indicator creation.

## Applied

Returns the volume type to apply.

```
ENUM_APPLIED_VOLUME Applied() const
```

### Returned value

Volume type to apply, defined at the indicator creation.

## Create

It creates the indicator with specified parameters. Use [Refresh\(\)](#) and [GetData\(\)](#) to update and get the indicator values.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period,        // Averaging period  
    ENUM_MA_METHOD  ma_method,        // Averaging method  
    ENUM_APPLIED_VOLUME applied       // Volume type to apply  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

*ma\_period*

[in] Averaging period.

*ma\_method*

[in] Averaging method ([ENUM\\_MA\\_METHOD](#) enumeration).

*applied*

[in] Volume type to apply ([ENUM\\_APPLIED\\_VOLUME](#) enumeration).

### Returned value

true if successful, false if indicator hasn't been created.

## Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

Buffer element of the specified index if successful, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Type

Returns the object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier ([IND\\_FORCE](#) for CiForce).

## CiMACD

CiMACD is a class intended for using the Moving Averages Convergence-Divergence technical indicator.

### Description

The CiMACD class provides the creation, setup and access to the data of the Moving Averages Convergence-Divergence indicator.

### Declaration

```
class CiMACD: public CIndicator
```

### Title

```
#include <Indicators\Oscillators.mqh>
```

### Class Methods

| Attributes                    |                                                 |
|-------------------------------|-------------------------------------------------|
| <a href="#">FastEmaPeriod</a> | Returns the averaging period for the fast EMA   |
| <a href="#">SlowEmaPeriod</a> | Returns the averaging period of the slow EMA    |
| <a href="#">SignalPeriod</a>  | Returns the averaging period of the signal line |
| <a href="#">Applied</a>       | Returns the price type or handle to apply       |
| Create Methods                |                                                 |
| <a href="#">Create</a>        | Creates the indicator                           |
| Data Access Methods           |                                                 |
| <a href="#">Main</a>          | Returns the buffer element of the main line     |
| <a href="#">Signal</a>        | Returns the buffer element of the signal line   |
| Input/output                  |                                                 |
| virtual <a href="#">Type</a>  | Returns the object type identifier              |



## FastEmaPeriod

Returns the averaging period for the fast EMA.

```
int FastEmaPeriod() const
```

### Returned value

Returns the averaging period for the fast EMA, defined at the indicator creation.

## SlowEmaPeriod

Returns the averaging period for the slow EMA.

```
int SlowEmaPeriod() const
```

### Returned value

Returns the averaging period for the slow EMA, defined at the indicator creation.

## SignalPeriod

Returns the averaging period for the signal line.

```
int SignalPeriod() const
```

### Returned value

Returns the averaging period for the signal line, defined at the indicator creation.

## Applied

Returns the price type or handle to apply.

```
int Applied() const
```

### Returned value

Price type or handle to apply, defined at the indicator creation.

## Create

It creates the indicator with specified parameters. Use [Refresh\(\)](#) and [GetData\(\)](#) to update and get the indicator values.

```
bool Create(  
    string          symbol,           // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             fast_ema_period,  // Fast EMA period  
    int             slow_ema_period,  // Slow EMA period  
    int             signal_period,    // Signal period  
    int             applied           // Price type or handle to apply  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

*fast\_ema\_period*

[in] Fast EMA period.

*slow\_ema\_period*

[in] Slow EMA period.

*signal\_period*

[in] Signal line period.

*applied*

[in] Price type or handle to apply.

### Returned value

true if successful, false if indicator hasn't been created.

## Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

Buffer element of the specified index if successful, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Signal

Returns the buffer element of the signal line by the specified index.

```
double Signal(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

The buffer element of the signal line of the specified index, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Type

Returns the object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier ([IND\\_MACD](#) for CiMACD).



## CiMomentum

CiMomentum is a class intended for using the Momentum technical indicator.

### Description

The CiMomentum class provides the creation, setup and access to the data of the Momentum indicator.

### Declaration

```
class CiMomentum: public CIndicator
```

### Title

```
#include <Indicators\Oscillators.mqh>
```

### Class Methods

| Attributes                   |                                           |
|------------------------------|-------------------------------------------|
| <a href="#">MaPeriod</a>     | Returns the averaging period              |
| <a href="#">Applied</a>      | Returns the price type or handle to apply |
| Create Methods               |                                           |
| <a href="#">Create</a>       | Creates the indicator                     |
| Data Access Methods          |                                           |
| <a href="#">Main</a>         | Returns the buffer element                |
| Input/output                 |                                           |
| virtual <a href="#">Type</a> | Returns the object type identifier        |

## MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

### Returned value

Returns the averaging period, defined at the indicator creation.

## Applied

Returns the price type or handle to apply.

```
int Applied() const
```

### Returned value

Price type or handle to apply, defined at the indicator creation.

## Create

It creates the indicator with specified parameters. Use [Refresh\(\)](#) and [GetData\(\)](#) to update and get the indicator values.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period,        // Averaging period  
    int             applied           // Price type or handle to apply  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

*ma\_period*

[in] Averaging period.

*applied*

[in] Price type or handle to apply.

### Returned value

true if successful, false if indicator hasn't been created.

## Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

Buffer element of the specified index if successful, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Type

Returns the object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier ([IND\\_MOMENTUM](#) for CiMomentum).

## CiOsMA

CiOsMA is a class intended for using the Moving Average of Oscillator (MACD histogram) technical indicator.

### Description

The CiOsMA class provides the creation, setup and access to the data of the Moving Average of Oscillator (MACD histogram) indicator.

### Declaration

```
class CiOsMA: public CIndicator
```

### Title

```
#include <Indicators\Oscilators.mqh>
```

### Class Methods

| Attributes                    |                                                 |
|-------------------------------|-------------------------------------------------|
| <a href="#">FastEmaPeriod</a> | Returns the averaging period of the fast EMA    |
| <a href="#">SlowEmaPeriod</a> | Returns the averaging period of the slow EMA    |
| <a href="#">SignalPeriod</a>  | Returns the averaging period of the signal line |
| <a href="#">Applied</a>       | Returns the price type or handle to apply       |
| Create Methods                |                                                 |
| <a href="#">Create</a>        | Creates the indicator                           |
| Data Access Methods           |                                                 |
| <a href="#">Main</a>          | Returns the buffer element                      |
| Input/output                  |                                                 |
| virtual <a href="#">Type</a>  | Returns the object type identifier              |

## FastEmaPeriod

Returns the averaging period for the fast EMA.

```
int FastEmaPeriod() const
```

### Returned value

Returns the averaging period for the fast EMA, defined at the indicator creation.



## SlowEmaPeriod

Returns the averaging period for the slow EMA.

```
int SlowEmaPeriod() const
```

### Returned value

Returns the averaging period for the slow EMA, defined at the indicator creation.

## SignalPeriod

Returns the averaging period for the signal line.

```
int SignalPeriod() const
```

### Returned value

Returns the averaging period for the signal line, defined at the indicator creation.

## Applied

Returns the price type or handle to apply.

```
int Applied() const
```

### Returned value

Price type or handle to apply, defined at the indicator creation.

## Create

It creates the indicator with specified parameters. Use [Refresh\(\)](#) and [GetData\(\)](#) to update and get the indicator values.

```
bool Create(  
    string          symbol,           // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             fast_ema_period,  // Fast EMA period  
    int             slow_ema_period,  // Slow EMA period  
    int             signal_period,    // Signal line period  
    int             applied           // Price type or handle to apply  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

*fast\_ema\_period*

[in] Fast EMA period.

*slow\_ema\_period*

[in] Slow EMA period.

*signal\_period*

[in] Signal line period.

*applied*

[in] Price type or handle to apply.

### Returned value

true if successful, false if indicator hasn't been created.

## Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

Buffer element of the specified index if successful, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Type

Returns the object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier ([IND\\_OSMA](#) for CiOsMA).

## CiRSI

CiRSI is a class intended for using the Relative Strength Index technical indicator.

### Description

The CiRSI class provides the creation, setup and access to the data of the Relative Strength Index indicator.

### Declaration

```
class CiRSI: public CIndicator
```

### Title

```
#include <Indicators\Oscillators.mqh>
```

### Class Methods

| Attributes                   |                                           |
|------------------------------|-------------------------------------------|
| <a href="#">MaPeriod</a>     | Returns the averaging period              |
| <a href="#">Applied</a>      | Returns the price type or handle to apply |
| Create Methods               |                                           |
| <a href="#">Create</a>       | Creates the indicator                     |
| Data Access Methods          |                                           |
| <a href="#">Main</a>         | Returns the buffer element                |
| Input/output                 |                                           |
| virtual <a href="#">Type</a> | Returns the object type identifier        |

## MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

### Returned value

Returns the averaging period, defined at the indicator creation.



## Applied

Returns the price type or handle to apply.

```
int Applied() const
```

### Returned value

Price type or handle to apply, defined at the indicator creation.

## Create

It creates the indicator with specified parameters. Use [Refresh\(\)](#) and [GetData\(\)](#) to update and get the indicator values.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period,        // Averaging period  
    int             applied           // Price type or handle to apply  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

*ma\_period*

[in] Averaging period.

*applied*

[in] Price type or handle to apply.

### Returned value

true if successful, false if indicator hasn't been created.

## Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

Buffer element of the specified index if successful, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Type

Returns the object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier ([IND\\_RSI](#) for CiRSI).

## CiRVI

CiRVI is a class intended for using the Relative Vigor Index technical indicator.

### Description

The CiRVI class provides the creation, setup and access to the data of the Relative Vigor Index indicator.

### Declaration

```
class CiRVI: public CIndicator
```

### Title

```
#include <Indicators\Oscillators.mqh>
```

### Class Methods

| Attributes                   |                                               |
|------------------------------|-----------------------------------------------|
| <a href="#">MaPeriod</a>     | Returns the averaging period                  |
| Create Methods               |                                               |
| <a href="#">Create</a>       | Creates the indicator                         |
| Data Access Methods          |                                               |
| <a href="#">Main</a>         | Returns the buffer element of the base line   |
| <a href="#">Signal</a>       | Returns the buffer element of the signal line |
| Input/output                 |                                               |
| virtual <a href="#">Type</a> | Returns the object type identifier            |

## MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

### Returned value

Returns the averaging period, defined at the indicator creation.

## Create

It creates the indicator with specified parameters. Use [Refresh\(\)](#) and [GetData\(\)](#) to update and get the indicator values.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period        // Averaging period  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

*ma\_period*

[in] Averaging period.

### Returned value

true if successful, false if indicator hasn't been created.

## Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

Buffer element of the specified index if successful, or [EMPTY\\_VALUE](#) if there isn't any correct data.



## Signal

Returns the buffer element of the signal line by the specified index.

```
double Signal(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

The buffer element of the signal line of the specified index, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Type

Returns the object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier ([IND\\_RVI](#) for CiRVI).

## CiStochastic

CiStochastic is a class intended for using the Stochastic Oscillator technical indicator.

### Description

The CiStochastic class provides the creation, setup and access to the data of the Stochastic Oscillator indicator.

### Declaration

```
class CiStochastic: public CIndicator
```

### Title

```
#include <Indicators\Oscillators.mqh>
```

### Class Methods

| Attributes                   |                                                |
|------------------------------|------------------------------------------------|
| <a href="#">Kperiod</a>      | Returns the averaging period for the %K line   |
| <a href="#">Dperiod</a>      | Returns the averaging period for the %D line   |
| <a href="#">Slowing</a>      | Returns the slowing period                     |
| <a href="#">MaMethod</a>     | Returns the averaging method                   |
| <a href="#">PriceField</a>   | Price type (Low/High ore Close/Close) to apply |
| <b>Create Methods</b>        |                                                |
| <a href="#">Create</a>       | Creates the indicator                          |
| <b>Data Access Methods</b>   |                                                |
| <a href="#">Main</a>         | Returns the buffer element of the base line    |
| <a href="#">Signal</a>       | Returns the buffer element of the signal line  |
| <b>Input/output</b>          |                                                |
| virtual <a href="#">Type</a> | Returns the object type identifier             |

## Kperiod

Returns the averaging period for the %K line.

```
int Kperiod() const
```

### Returned value

Returns the averaging period for the %K line, defined at the indicator creation.

## Dperiod

Returns the averaging period for the %D line.

```
int Dperiod() const
```

### Returned value

Returns the averaging period for the %D line, defined at the indicator creation.

## Slowing

Returns the period of slowing.

```
int Slowing() const
```

### Returned value

Returns the period of slowing, defined at the indicator creation.

## MaMethod

Returns the averaging method.

```
ENUM_MA_METHOD MaMethod() const
```

### Returned value

Returns the averaging method, defined at the indicator creation.

## PriceField

Returns the price type (Low/High or Close/Close) to apply.

```
ENUM_STO_PRICE PriceField() const
```

### Returned value

The price type to apply, defined at the indicator creation.



## Create

It creates the indicator with specified parameters. Use [Refresh\(\)](#) and [GetData\(\)](#) to update and get the indicator values.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             Kperiod,         // Averaging period of %K  
    int             Dperiod,         // Averaging period of %D  
    int             slowing,         // Slowing period  
    ENUM_MA_METHOD  ma_method,       // Averaging method  
    ENUM_STO_PRICE  price_field      // Price type to apply  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

*Kperiod*

[in] Averaging period of %K line.

*Dperiod*

[in] Averaging period of %D line.

*slowing*

[in] Slowing period.

*ma\_method*

[in] Averaging method ([ENUM\\_MA\\_METHOD](#) enumeration).

*price\_field*

[in] Price type (Low/High or Close/Close) to apply ([ENUM\\_STO\\_PRICE](#) enumeration).

### Returned value

true if successful, false if indicator hasn't been created.

## Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

Buffer element of the specified index if successful, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Signal

Returns the buffer element of the signal line by the specified index.

```
double Signal(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

## Signal

Returns the buffer element of the signal line by the specified index.

```
double Signal(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

The buffer element of the signal line of the specified index, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Type

Returns the object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier ([IND\\_STOCHASTIC](#) for CiStochastic).

## CiTriX

CiTriX is a class intended for using the Triple Exponential Moving Averages Oscillator technical indicator.

### Description

The CiTriX class provides the creation, setup and access to the data of the Triple Exponential Moving Averages Oscillator indicator.

### Declaration

```
class CiTriX: public CIndicator
```

### Title

```
#include <Indicators\Oscilators.mqh>
```

### Class Methods

| Attributes                   |                                           |
|------------------------------|-------------------------------------------|
| <a href="#">MaPeriod</a>     | Returns the averaging period              |
| <a href="#">Applied</a>      | Returns the price type or handle to apply |
| Create Methods               |                                           |
| <a href="#">Create</a>       | Creates the indicator                     |
| Data Access Methods          |                                           |
| <a href="#">Main</a>         | Returns the buffer element                |
| Input/output                 |                                           |
| virtual <a href="#">Type</a> | Returns the object type identifier        |

## MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

### Returned value

Returns the averaging period, defined at the indicator creation.

## Applied

Returns the price type or handle to apply.

```
int Applied() const
```

### Returned value

Price type or handle to apply, defined at the indicator creation.

## Create

It creates the indicator with specified parameters. Use [Refresh\(\)](#) and [GetData\(\)](#) to update and get the indicator values.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period,        // Averaging period  
    int             applied           // Price type or handle  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

*ma\_period*

[in] Averaging period.

*applied*

[in] Price type of handle to apply.

### Returned value

true if successful, false if indicator hasn't been created.



## Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

Buffer element of the specified index if successful, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Type

Returns the object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier ([IND\\_TRIX](#) for CiTriX).

## CiWPR

CiWPR is a class intended for using the Williams' Percent Range technical indicator.

### Description

The CiWPR class provides the creation, setup and access to the data of the Williams' Percent Range indicator.

### Declaration

```
class CiWPR: public CIndicator
```

### Title

```
#include <Indicators\Oscillators.mqh>
```

### Class Methods

| Attributes                   |                                    |
|------------------------------|------------------------------------|
| <a href="#">CalcPeriod</a>   | Returns the calculation period     |
| Create Methods               |                                    |
| <a href="#">Create</a>       | Creates the indicator              |
| Data Access Methods          |                                    |
| <a href="#">Main</a>         | Returns the buffer element         |
| Input/output                 |                                    |
| virtual <a href="#">Type</a> | Returns the object type identifier |

## CalcPeriod

Returns the period for calculation.

```
int CalcPeriod() const
```

### Returned value

Returns the period for calculation, defined at the indicator creation.

## Create

It creates the indicator with specified parameters. Use [Refresh\(\)](#) and [GetData\(\)](#) to update and get the indicator values.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             calc_period      // Calculation period  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

*calc\_period*

[in] Period for calculation.

### Returned value

true if successful, false if indicator hasn't been created.

## Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

Buffer element of the specified index if successful, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Type

Returns the object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier ([IND\\_WPR](#) for CiWPR).

## Volume Indicators

This group of chapters contains technical details of Volume Indicator classes of the MQL5 Standard Library and descriptions of all its key components.

| Class/group                      | Description               |
|----------------------------------|---------------------------|
| <a href="#"><u>CiAD</u></a>      | Accumulation/Distribution |
| <a href="#"><u>CiMFI</u></a>     | Money Flow Index          |
| <a href="#"><u>CiOBV</u></a>     | On Balance Volume         |
| <a href="#"><u>CiVolumes</u></a> | Volumes                   |



## CiAD

CiAD is a class intended for using the Accumulation/Distribution technical indicator.

### Description

The CiAD class provides the creation, setup and access to the data of the Accumulation/Distribution indicator.

### Declaration

```
class CiAD: public CIndicator
```

### Title

```
#include <Indicators\Volumes.mqh>
```

### Class Methods

| Attributes                   |                                    |
|------------------------------|------------------------------------|
| <a href="#">Applied</a>      | Returns the volume type to apply   |
| Create Methods               |                                    |
| <a href="#">Create</a>       | Creates the indicator              |
| Data Access Methods          |                                    |
| <a href="#">Main</a>         | Returns the buffer element         |
| Input/output                 |                                    |
| virtual <a href="#">Type</a> | Returns the object type identifier |

## Applied

Returns the volume type to apply.

```
ENUM_APPLIED_VOLUME Applied() const
```

### Returned value

Volume type to apply, defined at the indicator creation.

## Create

It creates the indicator with specified parameters. Use [Refresh\(\)](#) and [GetData\(\)](#) to update and get the indicator values.

```
bool Create(  
    string          symbol,      // Symbol  
    ENUM_TIMEFRAMES period,      // Period  
    ENUM_APPLIED_VOLUME applied  // Volume type to apply  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

*applied*

[in] Volume type to apply ([ENUM\\_APPLIED\\_VOLUME](#) enumeration).

### Returned value

true if successful, false if indicator hasn't been created.

## Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

Buffer element of the specified index if successful, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Type

Returns the object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier ([IND\\_AD](#) for CiAD).

## CiMFI

CiMFI is a class intended for using the Money Flow Index technical indicator.

### Description

The CiMFI class provides the creation, setup and access to the data of the Money Flow Index indicator.

### Declaration

```
class CiMFI: public CIndicator
```

### Title

```
#include <Indicators\Volumes.mqh>
```

### Class Methods

| Attributes                   |                                    |
|------------------------------|------------------------------------|
| <a href="#">MaPeriod</a>     | Returns the averaging period       |
| <a href="#">Applied</a>      | Returns the volume type to apply   |
| Create Methods               |                                    |
| <a href="#">Create</a>       | Creates the indicator              |
| Data Access Methods          |                                    |
| <a href="#">Main</a>         | Returns the buffer element         |
| Input/output                 |                                    |
| virtual <a href="#">Type</a> | Returns the object type identifier |

## MaPeriod

Returns the averaging period.

```
int MaPeriod() const
```

### Returned value

Returns the averaging period, defined at the indicator creation.

## Applied

Returns the volume type to apply.

```
ENUM_APPLIED_VOLUME Applied() const
```

### Returned value

Volume type to apply, defined at the indicator creation.



## Create

It creates the indicator with specified parameters. Use [Refresh\(\)](#) and [GetData\(\)](#) to update and get the indicator values.

```
bool Create(  
    string          symbol,          // Symbol  
    ENUM_TIMEFRAMES period,          // Period  
    int             ma_period,        // Averaging period  
    ENUM_APPLIED_VOLUME applied      // Volume type to apply  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

*ma\_period*

[in] Averaging period.

*applied*

[in] Volume type to apply ([ENUM\\_APPLIED\\_VOLUME](#) enumeration).

### Returned value

true if successful, false if indicator hasn't been created.

## Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

Buffer element of the specified index if successful, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Type

Returns the object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier ([IND\\_MFI](#) for CiMFI).

## CiOBV

CiOBV is a class intended for using the On Balance Volume technical indicator.

### Description

The CiOBV class provides the creation, setup and access to the data of the On Balance Volume indicator.

### Declaration

```
class CiOBV: public CIndicator
```

### Title

```
#include <Indicators\Volumes.mqh>
```

### Class Methods

| Attributes                   |                                    |
|------------------------------|------------------------------------|
| <a href="#">Applied</a>      | Returns the volume type to apply   |
| Create Methods               |                                    |
| <a href="#">Create</a>       | Creates the indicator              |
| Data Access Methods          |                                    |
| <a href="#">Main</a>         | Returns the buffer element         |
| Input/output                 |                                    |
| virtual <a href="#">Type</a> | Returns the object type identifier |

## Applied

Returns the volume type to apply.

```
ENUM_APPLIED_VOLUME Applied() const
```

### Returned value

Volume type to apply, defined at the indicator creation.

## Create

It creates the indicator with specified parameters. Use [Refresh\(\)](#) and [GetData\(\)](#) to update and get the indicator values.

```
bool Create(  
    string          symbol,      // Symbol  
    ENUM_TIMEFRAMES period,      // Period  
    ENUM_APPLIED_VOLUME applied  // Volume type to apply  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

*applied*

[in] Volume type to apply ([ENUM\\_APPLIED\\_VOLUME](#) enumeration).

### Returned value

true if successful, false if indicator hasn't been created.

## Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

Buffer element of the specified index if successful, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Type

Returns the object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier ([IND\\_OBV](#) for CiOBV).



## CiVolumes

CiVolumes is a class intended for using the Volumes technical indicator.

### Description

The CiVolumes class provides the creation, setup and access to the data of the Volumes indicator.

### Declaration

```
class CiVolumes: public CIndicator
```

### Title

```
#include <Indicators\Volumes.mqh>
```

### Class Methods

| Attributes                   |                                    |
|------------------------------|------------------------------------|
| <a href="#">Applied</a>      | Returns the volume type to apply   |
| Create Methods               |                                    |
| <a href="#">Create</a>       | Creates the indicator              |
| Data Access Methods          |                                    |
| <a href="#">Main</a>         | Returns the buffer element         |
| Input/output                 |                                    |
| virtual <a href="#">Type</a> | Returns the object type identifier |

## Applied

Returns the volume type to apply.

```
ENUM_APPLIED_VOLUME Applied() const
```

### Returned value

Volume type to apply, defined at the indicator creation.

## Create

It creates the indicator with specified parameters. Use [Refresh\(\)](#) and [GetData\(\)](#) to update and get the indicator values.

```
bool Create(  
    string          symbol,      // Symbol  
    ENUM_TIMEFRAMES period,      // Period  
    ENUM_APPLIED_VOLUME applied  // Volume type to apply  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

*applied*

[in] Volume type to apply ([ENUM\\_APPLIED\\_VOLUME](#) enumeration).

### Returned value

true if successful, false if indicator hasn't been created.

## Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

Buffer element of the specified index if successful, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Type

Returns the object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier ([IND\\_VOLUMES](#) for CiVolumes).

## Bill Williams Indicators

This group of chapters contains technical details of Bill Williams Indicator classes of the MQL5 Standard Library and descriptions of all its key components.

| Class/group                        | Description               |
|------------------------------------|---------------------------|
| <a href="#"><u>CiAC</u></a>        | Accelerator Oscillator    |
| <a href="#"><u>CiAlligator</u></a> | Alligator                 |
| <a href="#"><u>CiAO</u></a>        | Awesome Oscillator        |
| <a href="#"><u>CiFractals</u></a>  | Fractals                  |
| <a href="#"><u>CiGator</u></a>     | Gator Oscillator          |
| <a href="#"><u>CiBWMFI</u></a>     | Market Facilitation Index |

## CiAC

CiAC is a class intended for using the Accelerator Oscillator technical indicator.

### Description

The CiAC class provides the creation, setup and access to the data of the Accelerator Oscillator indicator.

### Declaration

```
class CiAC: public CIndicator
```

### Title

```
#include <Indicators\BillWilliams.mqh>
```

### Class Methods

| Create Methods               |                                    |
|------------------------------|------------------------------------|
| <a href="#">Create</a>       | Creates the indicator              |
| Data Access Methods          |                                    |
| <a href="#">Main</a>         | Returns the buffer element         |
| Input/output                 |                                    |
| virtual <a href="#">Type</a> | Returns the object type identifier |

## Create

It creates the indicator with specified parameters. Use [Refresh\(\)](#) and [GetData\(\)](#) to update and get the indicator values.

```
bool Create(  
    string      symbol,      // Symbol  
    ENUM_TIMEFRAMES period   // Period  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

### Returned value

true if successful, false if indicator hasn't been created.



## Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

Buffer element of the specified index if successful, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Type

Returns the object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier ([IND\\_AC](#) for CiAC).

## CiAlligator

CiAlligator is a class intended for using the Alligator technical indicator.

### Description

The CiAlligator class provides the creation, setup and access to the data of the Alligator indicator.

### Declaration

```
class CiAlligator: public CIndicator
```

### Title

```
#include <Indicators\BillWilliams.mqh>
```

### Class Methods

| Attributes                          |                                                      |
|-------------------------------------|------------------------------------------------------|
| <a href="#"><u>JawPeriod</u></a>    | Returns the averaging period for the Jaws line       |
| <a href="#"><u>JawShift</u></a>     | Returns the horizontal shift of the Jaws line        |
| <a href="#"><u>TeethPeriod</u></a>  | Returns the averaging period for the Teeths line     |
| <a href="#"><u>TeethShift</u></a>   | Returns the horizontal shift of the Teeths line      |
| <a href="#"><u>LipsPeriod</u></a>   | Returns the averaging period for the Lips line       |
| <a href="#"><u>LipsShift</u></a>    | Returns the horizontal shift of the Lips line        |
| <a href="#"><u>MaMethod</u></a>     | Returns the averaging method                         |
| <a href="#"><u>Applied</u></a>      | Returns the price type or handle to apply            |
| <b>Create Methods</b>               |                                                      |
| <a href="#"><u>Create</u></a>       | Creates the indicator                                |
| <b>Data Access Methods</b>          |                                                      |
| <a href="#"><u>Jaw</u></a>          | Returns the buffer element of the Jaws line buffer   |
| <a href="#"><u>Teeth</u></a>        | Returns the buffer element of the Teeths line buffer |
| <a href="#"><u>Lips</u></a>         | Returns the buffer element of the Lips line buffer   |
| <b>Input/output</b>                 |                                                      |
| virtual <a href="#"><u>Type</u></a> | Returns the object type identifier                   |

## JawPeriod

Returns the averaging period for the Jaw line.

```
int  JawPeriod() const
```

### Returned value

Returns the averaging period for the Jaw line, defined at the indicator creation.

## JawShift

Returns the horizontal shift of the Jaws line.

```
int  JawShift () const
```

### Returned value

Horizontal shift of the Jaws line, defined at the indicator creation.

## TeethPeriod

Returns the averaging period for the Teeth line.

```
int TeethPeriod() const
```

### Returned value

Returns the averaging period for the Teeth line, defined at the indicator creation.

## TeethShift

Returns the horizontal shift of the Teeths line.

```
int TeethShift() const
```

### Returned value

Horizontal shift of the Teeths line, defined at the indicator creation.

## LipsPeriod

Returns the averaging period for the Lips line.

```
int LipsPeriod() const
```

### Returned value

Returns the averaging period for the Lips line, defined at the indicator creation.



## LipsShift

Returns the horizontal shift of the Lips line.

```
int LipsShift() const
```

### Returned value

Horizontal shift of the Lips line, defined at the indicator creation.

## MaMethod

Returns the averaging method.

```
ENUM_MA_METHOD MaMethod() const
```

### Returned value

Returns the averaging method, defined at the indicator creation.

## Applied

Returns the price type or handle to apply.

```
int Applied() const
```

### Returned value

Price type or handle to apply, defined at the indicator creation.

## Create

It creates the indicator with specified parameters. Use [Refresh\(\)](#) and [GetData\(\)](#) to update and get the indicator values.

```
bool Create(
    string          symbol,           // Symbol
    ENUM_TIMEFRAMES period,          // Period
    int             jaw_period,       // Jaws period
    int             jaw_shift,        // Jaws shift
    int             teeth_period,     // Teeths period
    int             teeth_shift,      // Teeths shift
    int             lips_period,      // Lips period
    int             lips_shift,       // Lips shift
    ENUM_MA_METHOD  ma_method,       // Averaging method
    int             applied           // Volume type to apply
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

*jaw\_period*

[in] Jaws period.

*jaw\_shift*

[in] Jaws shift.

*teeth\_period*

[in] Teeths period.

*teeth\_shift*

[in] Teeths shift.

*lips\_period*

[in] Lips period.

*lips\_shift*

[in] Lips shift.

*ma\_method*

[in] Moving average method ([ENUM\\_MA\\_METHOD](#) enumeration).

*applied*

[in] Volume type to apply.

### Returned value

true if successful, false if indicator hasn't been created.

## Jaw

Returns the buffer element of the Jaws line by the specified index.

```
double Jaw(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

The buffer element of the Jaws line of the specified index, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Teeth

Returns the buffer element of the Teeths line by the specified index.

```
double Teeth(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

The buffer element of the Teeths line of the specified index, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Lips

Returns the buffer element of the Lips line by the specified index.

```
double Lips(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

The buffer element of the Lips line of the specified index, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Type

Returns the object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier ([IND\\_ALLIGATOR](#) for CiAlligator).



## CiAO

CiAO is a class intended for using the Awesome Oscillator technical indicator.

### Description

The CiAO class provides the creation, setup and access to the data of the Awesome Oscillator indicator.

### Declaration

```
class CiAO: public CIndicator
```

### Title

```
#include <Indicators\BillWilliams.mqh>
```

### Class Methods

| Create Methods               |                                    |
|------------------------------|------------------------------------|
| <a href="#">Create</a>       | Creates the indicator              |
| Data Access Methods          |                                    |
| <a href="#">Main</a>         | Returns the buffer element         |
| Input/output                 |                                    |
| virtual <a href="#">Type</a> | Returns the object type identifier |

## Create

It creates the indicator with specified parameters. Use [Refresh\(\)](#) and [GetData\(\)](#) to update and get the indicator values.

```
bool Create(  
    string      symbol,      // Symbol  
    ENUM_TIMEFRAMES period   // Period  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

### Returned value

true if successful, false if indicator hasn't been created.

## Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

Buffer element of the specified index if successful, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Type

Returns the object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier ([IND\\_AO](#) for CiAO).

## CiFractals

CiFractals is a class intended for using the Fractals technical indicator.

### Description

The CiFractals class provides the creation, setup and access to the data of the Fractals indicator.

### Declaration

```
class CiFractals: public CIndicator
```

### Title

```
#include <Indicators\BillWilliams.mqh>
```

### Class Methods

| Create Methods               |                                                |
|------------------------------|------------------------------------------------|
| <a href="#">Create</a>       | Creates the indicator                          |
| Data Access Methods          |                                                |
| <a href="#">Upper</a>        | Returns the buffer element of the upper buffer |
| <a href="#">Lower</a>        | Returns the buffer element of the lower buffer |
| Input/output                 |                                                |
| virtual <a href="#">Type</a> | Returns the object type identifier             |

## Create

It creates the indicator with specified parameters. Use [Refresh\(\)](#) and [GetData\(\)](#) to update and get the indicator values.

```
bool Create(  
    string      symbol,      // Symbol  
    ENUM_TIMEFRAMES period   // Period  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

### Returned value

true if successful, false if indicator hasn't been created.

## Upper

Returns the buffer element of the upper buffer by the specified index.

```
double Upper (  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

The buffer element of the upper buffer of the specified index, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Lower

Returns the buffer element of the lower buffer by the specified index.

```
double Lower (  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

The buffer element of the lower buffer of the specified index, or [EMPTY\\_VALUE](#) if there isn't any correct data.



## Type

Returns the object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier ([IND\\_FRACTALS](#) for CiFractals).

## CiGator

CiGator is a class intended for using the Gator Oscillator technical indicator.

### Description

The CiGator class provides the creation, setup and access to the data of the Gator Oscillator indicator.

### Declaration

```
class CiGator: public CIndicator
```

### Title

```
#include <Indicators\BillWilliams.mqh>
```

### Class Methods

| Attributes                   |                                                  |
|------------------------------|--------------------------------------------------|
| <a href="#">JawPeriod</a>    | Returns the averaging period for the Jaws line   |
| <a href="#">JawShift</a>     | Returns the horizontal shift of the Jaws line    |
| <a href="#">TeethPeriod</a>  | Returns the averaging period for the Teeths line |
| <a href="#">TeethShift</a>   | Returns the horizontal shift of the Teeths line  |
| <a href="#">LipsPeriod</a>   | Returns the averaging period for the Lips line   |
| <a href="#">LipsShift</a>    | Returns the horizontal shift of the Lips line    |
| <a href="#">MaMethod</a>     | Returns the averaging method                     |
| <a href="#">Applied</a>      | Returns the price type or handle to apply        |
| <b>Create Methods</b>        |                                                  |
| <a href="#">Create</a>       | Creates the indicator                            |
| <b>Data Access Methods</b>   |                                                  |
| <a href="#">Upper</a>        | Returns the buffer element of the upper buffer   |
| <a href="#">Lower</a>        | Returns the buffer element of the lower buffer   |
| <b>Input/output</b>          |                                                  |
| virtual <a href="#">Type</a> | Returns the object type identifier               |

## JawPeriod

Returns the averaging period for the Jaws line.

```
int  JawPeriod() const
```

### Returned value

Returns the averaging period for the Jaws line, defined at the indicator creation.

## JawShift

Returns the horizontal shift of the Jaws line.

```
int  JawShift () const
```

### Returned value

Horizontal shift of the Jaws line, defined at the indicator creation.

## TeethPeriod

Returns the averaging period for the Teeth line.

```
int TeethPeriod() const
```

### Returned value

Returns the averaging period for the Teeth line, defined at the indicator creation.

## TeethShift

Returns the horizontal shift of the Teeths line.

```
int TeethShift() const
```

### Returned value

Horizontal shift of the Teeths line, defined at the indicator creation.

## LipsPeriod

Returns the averaging period for the Lips line.

```
int LipsPeriod() const
```

### Returned value

Returns the averaging period for the Lips line, defined at the indicator creation.

## LipsShift

Returns the horizontal shift of the Lips line.

```
int LipsShift() const
```

### Returned value

Horizontal shift of the Lips line, defined at the indicator creation.



## MaMethod

Returns the averaging method.

```
ENUM_MA_METHOD MaMethod() const
```

### Returned value

Returns the averaging method, defined at the indicator creation.

## Applied

Returns the price type or handle to apply.

```
int Applied() const
```

### Returned value

Price type or handle to apply, defined at the indicator creation.

## Create

It creates the indicator with specified parameters. Use [Refresh\(\)](#) and [GetData\(\)](#) to update and get the indicator values.

```
bool Create(
    string          symbol,           // Symbol
    ENUM_TIMEFRAMES period,          // Period
    int             jaw_period,       // Jaws period
    int             jaw_shift,        // Jaws shift
    int             teeth_period,     // Teeths period
    int             teeth_shift,      // Teeths shift
    int             lips_period,      // Lips period
    int             lips_shift,       // Lips shift
    ENUM_MA_METHOD  ma_method,       // Averaging method
    int             applied           // Volume type to apply
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

*jaw\_period*

[in] Jaws period.

*jaw\_shift*

[in] Jaws shift.

*teeth\_period*

[in] Teeths period.

*teeth\_shift*

[in] Teeths shift.

*lips\_period*

[in] Lips period.

*lips\_shift*

[in] Lips shift.

*ma\_method*

[in] Averaging method ([ENUM\\_MA\\_METHOD](#) enumeration).

*applied*

[in] Volume type to apply.

### Returned value

true if successful, false if indicator hasn't been created.

## Upper

Returns the buffer element of the upper buffer by the specified index.

```
double Upper (  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

The buffer element of the upper buffer of the specified index, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Lower

Returns the buffer element of the lower buffer by the specified index.

```
double Upper (  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

The buffer element of the lower buffer of the specified index, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Type

Returns the object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier ([IND\\_GATOR](#) for CiGator).

## CiBWMFI

CiBWMFI is a class intended for using the Market Facilitation Index by Bill Williams technical indicator.

### Description

The CiBWMFI class provides the creation, setup and access to the data of the Market Facilitation Index by Bill Williams indicator.

### Declaration

```
class CiBWMFI: public CIndicator
```

### Title

```
#include <Indicators\BillWilliams.mqh>
```

### Class Methods

| Attributes                   |                                    |
|------------------------------|------------------------------------|
| <a href="#">Applied</a>      | Returns the volume type to apply   |
| Create Methods               |                                    |
| <a href="#">Create</a>       | Creates the indicator              |
| Data Access Methods          |                                    |
| <a href="#">Main</a>         | Returns the buffer element         |
| Input/output                 |                                    |
| virtual <a href="#">Type</a> | Returns the object type identifier |

## Applied

Returns the volume type to apply.

```
ENUM_APPLIED_VOLUME Applied() const
```

### Returned value

Volume type to apply, defined at the indicator creation.



## Create

It creates the indicator with specified parameters. Use [Refresh\(\)](#) and [GetData\(\)](#) to update and get the indicator values.

```
bool Create(  
    string          symbol,      // Symbol  
    ENUM_TIMEFRAMES period,      // Period  
    ENUM_APPLIED_VOLUME applied  // Volume type to apply  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

*applied*

[in] Volume type to apply ([ENUM\\_APPLIED\\_VOLUME](#) enumeration).

### Returned value

true if successful, false if indicator hasn't been created.

## Main

Returns the buffer element by the specified index.

```
double Main(  
    int index    // Index  
)
```

### Parameters

*index*

[in] Element index.

### Returned value

Buffer element of the specified index if successful, or [EMPTY\\_VALUE](#) if there isn't any correct data.

## Type

Returns the object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier ([IND\\_BWMFI](#) for CiBWMFI).

## CiCustom

CiCustom is a class intended for using the custom technical indicators.

### Description

The CiCustom class provides the creation, setup and access to the data of the custom indicator.

### Declaration

```
class CiCustom: public CIndicator
```

### Title

```
#include <Indicators\Custom.mqh>
```

### Class Methods

| Attributes                   |                                                           |
|------------------------------|-----------------------------------------------------------|
| <a href="#">NumBuffers</a>   | Sets the number of buffers                                |
| <a href="#">NumParams</a>    | Gets the number of parameters                             |
| <a href="#">ParamType</a>    | Gets the type of the specified parameter                  |
| <a href="#">ParamLong</a>    | Gets the value of the specified parameter of integer type |
| <a href="#">ParamDouble</a>  | Gets the value of the specified parameter of double type  |
| <a href="#">ParamString</a>  | Gets the value of the specified parameter of string type  |
| Input/output                 |                                                           |
| virtual <a href="#">Type</a> | Gets the object type identifier                           |

## NumBuffers

Sets the number of buffers.

```
bool NumBuffers(  
    int buffers    // number of buffers  
)
```

### Returned value

true if successful, false if buffers haven't been set.

## NumParams

Gets the number of parameters.

```
int NumParams() const
```

### Returned value

Number of parameters, used in creation of the indicator.

## ParamType

Gets a type of the parameter with specified index.

```
ENUM_DATATYPE ParamType(  
    int index    // parameter index  
) const
```

### Parameters

*index*

[in] Parameter index.

### Returned value

Returns the data type (value of [ENUM\\_DATATYPE](#) enumeration) of the parameter with specified index, used in indicator creation.

### Note

If parameter index is invalid, it returns [WRONG\\_VALUE](#).

## ParamLong

Gets the value of specified parameter of long type.

```
long ParamLong(  
    int index    // index  
    ) const
```

### Parameters

*index*

[in] Parameter index.

### Returned value

The value of specified parameter of long type, used in creation of the indicator.

### Note

If the number is invalid, it returns 0.



## ParamDouble

Gets the value of specified parameter of double type.

```
double ParamDouble(  
    int index    // index  
) const
```

### Parameters

*index*

[in] Parameter index.

### Returned value

The value of specified parameter of double type, used in creation of the indicator.

### Note

If the number is invalid, it returns [EMPTY\\_VALUE](#).

## ParamString

Gets the value of specified parameter of string type.

```
string ParamString(  
    int index    // index  
    ) const
```

### Parameters

*index*

[in] Parameter index.

### Returned value

The value of specified string parameter, used in creation of the indicator.

### Note

If the number is invalid, it returns an empty string.

## Type

Returns the object type identifier.

```
virtual int Type() const
```

### Returned value

Object type identifier ([IND\\_CUSTOM](#) for CiCustom).

## Trade Classes

This section contains technical details of working with trade classes and description of the relevant components of the MQL5 standard library.

Using trade classes will save time when creating custom programs (Expert Advisors).

MQL5 Standard Library (in terms of data sets) is placed in the terminal working directory, in the Include\Arrays folder.

| Class/Group                              | Description                                                  |
|------------------------------------------|--------------------------------------------------------------|
| <a href="#"><u>CAccountInfo</u></a>      | Class for working with trade account properties              |
| <a href="#"><u>CSymbolInfo</u></a>       | Class for working with trade instrument properties           |
| <a href="#"><u>COrderInfo</u></a>        | Class for working with pending order properties              |
| <a href="#"><u>CHistoryOrderInfo</u></a> | Class for working with history order properties              |
| <a href="#"><u>CPositionInfo</u></a>     | Class for working with open position properties              |
| <a href="#"><u>CDealInfo</u></a>         | Class for working with history deal properties               |
| <a href="#"><u>CTrade</u></a>            | Class for trade operations execution                         |
| <a href="#"><u>CTerminalInfo</u></a>     | Class for getting the properties of mql5 program environment |

## CAccountInfo

CAccountInfo is a class for easy access to the currently opened trade account properties.

### Description

CAccountInfo class provides easy access to the currently opened trade account properties.

### Declaration

```
class CAccountInfo : public CObject
```

### Title

```
#include <Trade\AccountInfo.mqh>
```

### Class methods by groups

| Access to integer type properties     |                                                   |
|---------------------------------------|---------------------------------------------------|
| <a href="#">Login</a>                 | Gets the account number                           |
| <a href="#">TradeMode</a>             | Gets the trade mode                               |
| <a href="#">TradeModeDescription</a>  | Gets the trade mode as a string                   |
| <a href="#">Leverage</a>              | Gets the amount of given leverage                 |
| <a href="#">MarginMode</a>            | Gets the mode of account stop out                 |
| <a href="#">MarginModeDescription</a> | Gets the description of account stop out mode     |
| <a href="#">TradeAllowed</a>          | Gets the flag of trade allowance                  |
| <a href="#">TradeExpert</a>           | Gets the flag of automated trade allowance        |
| <a href="#">LimitOrders</a>           | Gets the maximal number of allowed pending orders |
| Access to double type properties      |                                                   |
| <a href="#">Balance</a>               | Gets the balance of account                       |
| <a href="#">Credit</a>                | Gets the amount of given credit                   |
| <a href="#">Profit</a>                | Gets the amount of current profit on account      |
| <a href="#">Equity</a>                | Gets the amount of current equity on account      |
| <a href="#">Margin</a>                | Gets the amount of reserved margin                |
| <a href="#">FreeMargin</a>            | Gets the amount of free margin                    |
| <a href="#">MarginLevel</a>           | Gets the level of margin                          |
| <a href="#">MarginCall</a>            | Gets the level of margin for deposit              |
| <a href="#">MarginStopOut</a>         | Gets the level of margin for Stop Out             |

|                                         |                                                                         |
|-----------------------------------------|-------------------------------------------------------------------------|
| <b>Access to text properties</b>        |                                                                         |
| <a href="#"><u>Name</u></a>             | Gets the client name                                                    |
| <a href="#"><u>Server</u></a>           | Gets the trade server name                                              |
| <a href="#"><u>Currency</u></a>         | Gets the deposit currency name                                          |
| <a href="#"><u>Company</u></a>          | Gets the company name, that serves an account                           |
| <b>Access to MQL5 API functions</b>     |                                                                         |
| <a href="#"><u>InfoInteger</u></a>      | Gets the value of specified integer type property                       |
| <a href="#"><u>InfoDouble</u></a>       | Gets the value of specified double type property                        |
| <a href="#"><u>InfoString</u></a>       | Gets value of specified string type property                            |
| <b>Additional methods</b>               |                                                                         |
| <a href="#"><u>OrderProfitCheck</u></a> | Gets the evaluated profit, based on the parameters passed               |
| <a href="#"><u>MarginCheck</u></a>      | Gets the amount of margin, required to execute trade operation          |
| <a href="#"><u>FreeMarginCheck</u></a>  | Gets the amount of free margin, left after execution of trade operation |
| <a href="#"><u>MaxLotCheck</u></a>      | Gets the maximal possible volume of trade operation                     |

## Login

Gets the account number.

```
long Login() const
```

### Returned value

Account number.

## TradeMode

Gets the trade mode.

```
ENUM_ACCOUNT_TRADE_MODE TradeMode() const
```

### Returned value

Trade mode (value of [ENUM\\_ACCOUNT\\_TRADE\\_MODE](#) enumeration).



## TradeModeDescription

Gets the trade mode as a string.

```
string TradeModeDescription() const
```

### Returned value

Trade mode as a string.

## Leverage

Gets the amount of given leverage.

```
long Leverage() const
```

### Returned value

Amount of given leverage.

## MarginMode

Gets the mode of account Stop Out.

```
ENUM_ACCOUNT_STOPOUT_MODE MarginMode() const
```

### Returned value

Account Stop Out mode (value of [ENUM\\_ACCOUNT\\_STOPOUT\\_MODE](#) enumeration).

## MarginModeDescription

Gets the mode of setting minimal margin level as a string.

```
string MarginModeDescription() const
```

### Returned value

Mode of setting minimal margin level as a string.

## TradeAllowed

Gets the flag of trade allowance.

```
bool TradeAllowed() const
```

### Returned value

Flag of trade allowance.

## TradeExpert

Gets the flag of automated trade allowance.

```
bool TradeExpert() const
```

### Returned value

Flag of automated trade allowance.

## LimitOrders

Gets the maximal number of allowed pending orders

```
int LimitOrders() const
```

### Returned value

The maximal number of allowed pending orders.

### Note

0 - no limits.

## Balance

Gets the balance of account.

```
double Balance() const
```

### Returned value

The balance of account (in deposit currency).



## Credit

Gets the amount of given credit.

```
double Credit() const
```

### Returned value

Amount of given credit (in deposit currency).

## Profit

Gets the amount of current profit on account.

```
double Profit() const
```

### Returned value

Amount of current profit on account (in deposit currency).

## Equity

Gets the amount of current equity on account.

```
double Equity() const
```

### Returned value

Amount of current equity on account (in deposit currency).

## Margin

Gets the amount of reserved margin.

```
double Margin() const
```

### Returned value

Amount of reserved margin (in deposit currency).

## FreeMargin

Gets the amount of free margin.

```
double FreeMargin() const
```

### Returned value

Amount of free margin (in deposit currency).

## MarginLevel

Gets the level of margin.

```
double MarginLevel() const
```

### Returned value

Level of margin.

## MarginCall

Gets the level of margin for a deposit.

```
double MarginCall() const
```

### Returned value

Level of margin for a deposit.

## MarginStopOut

Gets the level of margin for Stop Out.

```
double MarginStopOut() const
```

### Returned value

Level of margin for Stop Out.



## Name

Gets the client name.

```
string Name() const
```

### Returned value

Client name.

## Server

Gets the trade server name.

```
string Server() const
```

### Returned value

Trade server name.

## Currency

Gets the deposit currency name.

```
string Currency() const
```

### Returned value

Deposit currency name.

## Company

Gets the company name, that serves an account.

```
string Company() const
```

### Returned value

Company name, that serves an account.

## InfoInteger

Gets the value of specified integer type property.

```
long InfoInteger(  
    ENUM_ACCOUNT_INFO_INTEGER prop_id    // property ID  
) const
```

### Parameters

*prop\_id*

[in] Identifier of the property. The value can be one of the values of [ENUM\\_ACCOUNT\\_INFO\\_INTEGER](#) enumeration.

### Returned value

Value of [long](#) type.

## InfoDouble

Gets the value of specified double type property.

```
double InfoDouble(  
    ENUM_ACCOUNT_INFO_DOUBLE prop_id // property ID  
) const
```

### Parameters

*prop\_id*

[in] Identifier of the property. The value can be one of the values of [ENUM\\_ACCOUNT\\_INFO\\_DOUBLE](#) enumeration.

### Returned value

Value of [double](#) type.

## InfoString

Gets the value of specified string type property.

```
string InfoString(  
    ENUM_ACCOUNT_INFO_STRING prop_id    // property ID  
) const
```

### Parameters

*prop\_id*

[in] Identifier of the property. The value can be one of the values of [ENUM\\_ACCOUNT\\_INFO\\_STRING](#) enumeration.

### Returned value

Value of [string](#) type.

## OrderProfitCheck

The function calculates the profit for the current account, based on the parameters passed. The function is used for pre-evaluation of the result of a trade operation. The value is returned in the account currency.

```
double OrderProfitCheck(  
    const string      symbol,           // symbol  
    ENUM_ORDER_TYPE   trade_operation,  // operation type (ORDER_TYPE_BUY or ORDER_TYPE_SELL)  
    double            volume,           // volume  
    double            price_open,       // open price  
    double            price_close       // close price  
) const
```

### Parameters

*symbol*

[in] Symbol for trade operation.

*trade\_operation*

[in] Type of trade operation ([ENUM\\_ORDER\\_TYPE](#) enumeration).

*volume*

[in] Volume of trade operation.

*price\_open*

[in] Open price.

*price\_close*

[in] Close price.

### Returned value

If successful, it returns amount of profit or [EMPTY\\_VALUE](#) in the case of error.



## MarginCheck

Gets the amount of margin, required for trade operation.

```
double MarginCheck(  
    const string      symbol,           // symbol  
    ENUM_ORDER_TYPE   trade_operation, // operation  
    double            volume,          // volume  
    double            price            // price  
) const
```

### Parameters

*symbol*

[in] Symbol for trade operation.

*trade\_operation*

[in] Type of trade operation ([ENUM\\_ORDER\\_TYPE](#) enumeration).

*volume*

[in] Volume of trade operation.

*price*

[in] Price of trade operation.

### Returned value

Amount of margin, required for trade operation.

## FreeMarginCheck

Gets the amount of free margin, left after trade operation.

```
double FreeMarginCheck(  
    const string      symbol,           // symbol  
    ENUM_ORDER_TYPE   trade_operation,  // operation  
    double            volume,          // volume  
    double            price            // price  
) const
```

### Parameters

*symbol*

[in] Symbol for trade operation.

*trade\_operation*

[in] Type of trade operation ([ENUM\\_ORDER\\_TYPE](#) enumeration).

*volume*

[in] Volume of trade operation.

*price*

[in] Price of trade operation.

### Returned value

Amount of free margin, left after trade operation.

## MaxLotCheck

Gets the maximum possible volume of trade operation.

```
double MaxLotCheck(  
    const string      symbol,           // symbol  
    ENUM_ORDER_TYPE   trade_operation,  // operation  
    double            price,           // price  
    double            percent=100       // percent of available margin (0-100%), u  
    ) const
```

### Parameters

*symbol*

[in] Symbol for trade operation.

*trade\_operation*

[in] Type of trade operation ([ENUM\\_ORDER\\_TYPE](#) enumeration).

*price*

[in] Price of trade operation.

*percent=100*

[in] Percent of available margin (0-100%), used for trade operation.

### Returned value

Maximum possible volume of trade operation.

## CSymbolInfo

CSymbolInfo is a class for easy access to the symbol properties.

### Description

CSymbolInfo class provides access to the symbol properties.

### Declaration

```
class CSymbolInfo : public CObject
```

### Title

```
#include <Trade\SymbolInfo.mqh>
```

### Class methods by groups

| Controlling                    |                                                            |
|--------------------------------|------------------------------------------------------------|
| <a href="#">Refresh</a>        | Refreshes the symbol data                                  |
| <a href="#">RefreshRates</a>   | Refreshes the symbol quotes                                |
| Properties                     |                                                            |
| <a href="#">Name</a>           | Gets/sets symbol name                                      |
| <a href="#">Select</a>         | Gets/sets the "Market Watch" symbol flag                   |
| <a href="#">IsSynchronized</a> | Checks the symbol synchronization with server              |
| Volumes                        |                                                            |
| <a href="#">Volume</a>         | Gets the volume of last deal                               |
| <a href="#">VolumeHigh</a>     | Gets the maximal volume for a day                          |
| <a href="#">VolumeLow</a>      | Gets the minimal volume for a day                          |
| Miscellaneous                  |                                                            |
| <a href="#">Time</a>           | Gets the time of last quote                                |
| <a href="#">Spread</a>         | Gets the amount of spread (in points)                      |
| <a href="#">SpreadFloat</a>    | Gets the flag of floating spread                           |
| <a href="#">TicksBookDepth</a> | Gets the depth of ticks saving                             |
| Levels                         |                                                            |
| <a href="#">StopsLevel</a>     | Gets the minimal indent for orders (in points)             |
| <a href="#">FreezeLevel</a>    | Gets the distance of freezing trade operations (in points) |
| Bid prices                     |                                                            |

|                                                     |                                                        |
|-----------------------------------------------------|--------------------------------------------------------|
| <a href="#"><u>Bid</u></a>                          | Gets the current Bid price                             |
| <a href="#"><u>BidHigh</u></a>                      | Gets the maximal Bid price for a day                   |
| <a href="#"><u>BidLow</u></a>                       | Gets the minimal Bid price for a day                   |
| <b>Ask prices</b>                                   |                                                        |
| <a href="#"><u>Ask</u></a>                          | Gets the current Ask price                             |
| <a href="#"><u>AskHigh</u></a>                      | Gets the maximal Ask price for a day                   |
| <a href="#"><u>AskLow</u></a>                       | Gets the minimal Ask price for a day                   |
| <b>Prices</b>                                       |                                                        |
| <a href="#"><u>Last</u></a>                         | Gets the current Last price                            |
| <a href="#"><u>LastHigh</u></a>                     | Gets the maximal Last price for a day                  |
| <a href="#"><u>LastLow</u></a>                      | Gets the minimal Last price for a day                  |
| <b>Trade modes</b>                                  |                                                        |
| <a href="#"><u>TradeCalcMode</u></a>                | Gets the mode of contract cost calculation             |
| <a href="#"><u>TradeCalcModeDescription</u></a>     | Gets the mode of contract cost calculation as a string |
| <a href="#"><u>TradeMode</u></a>                    | Gets the type of order execution                       |
| <a href="#"><u>TradeModeDescription</u></a>         | Gets the type of order execution as a string           |
| <a href="#"><u>TradeExecution</u></a>               | Gets the closing of deals mode                         |
| <a href="#"><u>TradeExecutionDescription</u></a>    | Gets the closing of deals mode as a string             |
| <b>Swaps</b>                                        |                                                        |
| <a href="#"><u>SwapMode</u></a>                     | Gets the swap calculation model                        |
| <a href="#"><u>SwapModeDescription</u></a>          | Gets the swap calculation model as a string            |
| <a href="#"><u>SwapRollover3days</u></a>            | Gets the day of triple swap charge                     |
| <a href="#"><u>SwapRollover3daysDescription</u></a> | Gets the day of triple swap charge as a string         |
| <b>Margins and flags</b>                            |                                                        |
| <a href="#"><u>MarginInitial</u></a>                | Gets the value of initial margin                       |
| <a href="#"><u>MarginMaintenance</u></a>            | Gets the value of maintenance margin                   |
| <a href="#"><u>MarginLong</u></a>                   | Gets the rate of margin charging for long positions    |
| <a href="#"><u>MarginShort</u></a>                  | Gets the rate of margin charging for short positions   |
| <a href="#"><u>MarginLimit</u></a>                  | Gets the rate of margin charging for Limit orders      |
| <a href="#"><u>MarginStop</u></a>                   | Gets the rate of margin charging for Stop              |

|                                        |                                                                                                              |
|----------------------------------------|--------------------------------------------------------------------------------------------------------------|
|                                        | orders                                                                                                       |
| <a href="#"><u>MarginStopLimit</u></a> | Gets the rate of margin charging for StopLimit orders                                                        |
| <a href="#"><u>TradeTimeFlags</u></a>  | Gets the flags of the order expiration allowed modes                                                         |
| <a href="#"><u>TradeFillFlags</u></a>  | Gets the flags of the order filling allowed modes                                                            |
| <b>Quantization</b>                    |                                                                                                              |
| <a href="#"><u>Digits</u></a>          | Gets the number of digits after period                                                                       |
| <a href="#"><u>Point</u></a>           | Gets the value of one point                                                                                  |
| <a href="#"><u>TickValue</u></a>       | Gets the cost of tick (minimal change of price)                                                              |
| <a href="#"><u>TickValueProfit</u></a> | Gets the calculated tick price for a profitable position                                                     |
| <a href="#"><u>TickValueLoss</u></a>   | Gets the calculated tick price for a losing position                                                         |
| <a href="#"><u>TickSize</u></a>        | Gets the minimal change of price                                                                             |
| <b>Contracts sizes</b>                 |                                                                                                              |
| <a href="#"><u>ContractSize</u></a>    | Gets the amount of trade contract                                                                            |
| <a href="#"><u>LotsMin</u></a>         | Gets the minimal volume to close a deal                                                                      |
| <a href="#"><u>LotsMax</u></a>         | Gets the maximal volume to close a deal                                                                      |
| <a href="#"><u>LotsStep</u></a>        | Gets the minimal step of volume change to close a deal                                                       |
| <a href="#"><u>LotsLimit</u></a>       | Gets the maximal allowed volume of opened position and pending orders (direction insensitive) for one symbol |
| <b>Swaps sizes</b>                     |                                                                                                              |
| <a href="#"><u>SwapLong</u></a>        | Gets the value of long position swap                                                                         |
| <a href="#"><u>SwapShort</u></a>       | Gets the value of short position swap                                                                        |
| <b>Text properties</b>                 |                                                                                                              |
| <a href="#"><u>CurrencyBase</u></a>    | Gets the name of symbol base currency                                                                        |
| <a href="#"><u>CurrencyProfit</u></a>  | Gets the profit currency name                                                                                |
| <a href="#"><u>CurrencyMargin</u></a>  | Gets the margin currency name                                                                                |
| <a href="#"><u>Bank</u></a>            | Gets the name of current quote source                                                                        |
| <a href="#"><u>Description</u></a>     | Gets the string description of symbol                                                                        |
| <a href="#"><u>Path</u></a>            | Gets the path in symbols tree                                                                                |
| <b>Symbol properties</b>               |                                                                                                              |

|                                                |                                                                    |
|------------------------------------------------|--------------------------------------------------------------------|
| <a href="#"><u>SessionDeals</u></a>            | Gets the number of deals in the current session                    |
| <a href="#"><u>SessionBuyOrders</u></a>        | Gets the number of Buy orders at the moment                        |
| <a href="#"><u>SessionSellOrders</u></a>       | Gets the number of Sell orders at the moment                       |
| <a href="#"><u>SessionTurnover</u></a>         | Gets the summary turnover of the current session                   |
| <a href="#"><u>SessionInterest</u></a>         | Gets the summary open interest of the current session              |
| <a href="#"><u>SessionBuyOrdersVolume</u></a>  | Gets the current volume of Buy orders                              |
| <a href="#"><u>SessionSellOrdersVolume</u></a> | Gets the current volume of Sell orders                             |
| <a href="#"><u>SessionOpen</u></a>             | Gets the open price of the current session                         |
| <a href="#"><u>SessionClose</u></a>            | Gets the close price of the current session                        |
| <a href="#"><u>SessionAW</u></a>               | Gets the average weighted price of the current session             |
| <a href="#"><u>SessionPriceSettlement</u></a>  | Gets the settlement price of the current session                   |
| <a href="#"><u>SessionPriceLimitMin</u></a>    | Gets the minimal price of the current session                      |
| <a href="#"><u>SessionPriceLimitMax</u></a>    | Gets the maximal price of the current session                      |
| <b>Access to MQL5 API functions</b>            |                                                                    |
| <a href="#"><u>InfoInteger</u></a>             | Gets the value of specified integer type property                  |
| <a href="#"><u>InfoDouble</u></a>              | Gets the value of specified double type property                   |
| <a href="#"><u>InfoString</u></a>              | Gets the value of specified string type property                   |
| <b>Service functions</b>                       |                                                                    |
| <a href="#"><u>NormalizePrice</u></a>          | Returns the value of price, normalized using the symbol properties |

## Refresh

Refreshes the symbol data.

```
void Refresh()
```

### Returned value

None.

### Note

The symbol should be selected by [Name](#) method.



## RefreshRates

Refreshes the symbol quotes data.

```
bool RefreshRates ()
```

### Returned value

true - in case of success, false - if unable to refresh quotes.

### Note

The symbol should be selected by [Name](#) method.

## Name

Gets symbol name.

```
string Name() const
```

### Returned value

Symbol name.

## Name

Sets symbol name.

```
bool Name(string name)
```

### Returned value

None.

## Select

Gets the "Market Watch" symbol flag.

```
bool Select() const
```

### Returned value

Gets the "Market Watch" symbol flag.

## Select

Sets the "Market Watch" symbol flag.

```
bool Select()
```

### Returned value

true - in case of success, false - if unable to change flag.

## IsSynchronized

Checks the symbol synchronization with server.

```
bool IsSynchronized() const
```

### Returned value

true - if the symbol is synchronized with server, false - if not.

### Note

The symbol should be selected by [Name](#) method.

## Volume

Gets the volume of last deal.

```
long Volume() const
```

### Returned value

Volume of last deal.

### Note

The symbol should be selected by [Name](#) method.

## VolumeHigh

Gets the maximal volume of the day.

```
long VolumeHigh() const
```

### Returned value

Maximal volume of the day.

### Note

The symbol should be selected by [Name](#) method.

## VolumeLow

Gets the minimal volume of the day.

```
long VolumeLow() const
```

### Returned value

Minimal volume of the day.

### Note

The symbol should be selected by [Name](#) method.

## Time

Gets the time of last quote.

```
datetime Time() const
```

### Returned value

Time of last quote.

### Note

The symbol should be selected by [Name](#) method.



## Spread

Gets the amount of spread (in points).

```
int Spread() const
```

### Returned value

Gets the amount of spread (in points).

### Note

The symbol should be selected by [Name](#) method.

## SpreadFloat

Gets the flag of floating spread.

```
bool SpreadFloat() const
```

### Returned value

Flag of floating spread.

### Note

The symbol should be selected by [Name](#) method.

## TicksBookDepth

Gets the depth of ticks saving.

```
int TicksBookDepth() const
```

### Returned value

Depth of ticks saving.

### Note

The symbol should be selected by [Name](#) method.

## StopsLevel

Gets the minimal stop level for orders (in points).

```
int StopsLevel() const
```

### Returned value

Minimal stop level for orders (in points).

### Note

The symbol should be selected by [Name](#) method.

## FreezeLevel

Gets the freeze level (in points).

```
int FreezeLevel() const
```

### Returned value

Distance of freeze level (in points).

### Note

The symbol should be selected by [Name](#) method.

## Bid

Gets the current Bid price.

```
double Bid() const
```

### Returned value

Current Bid price.

### Note

The symbol should be selected by [Name](#) method.

## BidHigh

Gets the maximal Bid price of the day.

```
double BidHigh() const
```

### Returned value

Maximal Bid price of the day.

### Note

The symbol should be selected by [Name](#) method.

## BidLow

Gets the minimal Bid price of the day.

```
double BidLow() const
```

### Returned value

Minimal Bid price of the day.

### Note

The symbol should be selected by [Name](#) method.



## Ask

Gets the current Ask price.

```
double Ask() const
```

### Returned value

Current Ask price.

### Note

The symbol should be selected by [Name](#) method.

## AskHigh

Gets the maximal Ask price for a day.

```
double AskHigh() const
```

### Returned value

Maximal Ask price of the day.

### Note

The symbol should be selected by [Name](#) method.

## AskLow

Gets the minimal Ask price for a day.

```
double AskLow() const
```

### Returned value

Minimal Ask price of the day.

### Note

The symbol should be selected by [Name](#) method.

## Last

Gets the current Last price.

```
double Last() const
```

### Returned value

Current Last price.

## LastHigh

Gets the maximal Last price of the day.

```
double LastHigh() const
```

### Returned value

Maximal Last price of the day.

### Note

The symbol should be selected by [Name](#) method.

## LastLow

Gets the minimal Last price of the day.

```
double LastLow() const
```

### Returned value

Minimal Last price of the day.

### Note

The symbol should be selected by [Name](#) method.

## TradeCalcMode

Gets the mode of contract cost calculation.

```
ENUM_SYMBOL_CALC_MODE TradeCalcMode() const
```

### Returned value

Mode of contract cost calculation (value of [ENUM\\_SYMBOL\\_CALC\\_MODE](#) enumeration).

### Note

The symbol should be selected by [Name](#) method.

## TradeCalcModeDescription

Gets the mode of contract cost calculation as a string.

```
string TradeCalcModeDescription() const
```

### Returned value

Mode of contract cost calculation as a string.

### Note

The symbol should be selected by [Name](#) method.



## TradeMode

Gets the order execution type.

```
ENUM_SYMBOL_TRADE_MODE TradeMode() const
```

### Returned value

Order execution type (value of [ENUM\\_SYMBOL\\_TRADE\\_MODE](#) enumeration).

### Note

The symbol should be selected by [Name](#) method.

## TradeModeDescription

Gets the trade mode as a string.

```
string TradeModeDescription() const
```

### Returned value

Trade mode as a string.

### Note

The symbol should be selected by [Name](#) method.

## TradeExecution

Gets the trade execution mode.

```
ENUM_SYMBOL_TRADE_EXECUTION TradeExecution() const
```

### Returned value

Trade execution mode (value of [ENUM\\_SYMBOL\\_TRADE\\_EXECUTION](#) enumeration).

### Note

The symbol should be selected by [Name](#) method.

## TradeExecutionDescription

Gets the description of trade execution mode as a string.

```
string TradeExecutionDescription() const
```

### Returned value

Trade execution mode as a string.

### Note

The symbol should be selected by [Name](#) method.

## SwapMode

Gets the swap calculation mode.

```
ENUM_SYMBOL_SWAP_MODE SwapMode() const
```

### Returned value

Swap calculation mode (value of [ENUM\\_SYMBOL\\_SWAP\\_MODE](#) enumeration).

### Note

The symbol should be selected by [Name](#) method.

## SwapModeDescription

Gets the swap mode description as a string.

```
string SwapModeDescription() const
```

### Returned value

Swap mode description as a string.

### Note

The symbol should be selected by [Name](#) method.

## SwapRollover3days

Gets the swap rollover day.

```
ENUM_DAY_OF_WEEK SwapRollover3days () const
```

### Returned value

Swap rollover day (value of [ENUM\\_DAY\\_OF\\_WEEK](#) enumeration).

### Note

The symbol should be selected by [Name](#) method.

## SwapRollover3daysDescription

Gets the swap rollover day as a string.

```
string SwapRollover3daysDescription() const
```

### Returned value

Swap rollover day as a string.

### Note

The symbol should be selected by [Name](#) method.



## MarginInitial

Gets the value of initial margin.

```
double MarginInitial ()
```

### Returned value

Value of initial margin.

### Note

It points the amount of margin (in margin currency of instrument), that is charged from one lot. Used to check client's equity, when he enters the market.

The symbol should be selected by [Name](#) method.

## MarginMaintenance

Gets the value of maintenance margin.

```
double MarginMaintenance()
```

### Returned value

Value of maintenance margin.

### Note

It points the amount of margin (in margin currency of instrument), that is charged from one lot. Used to check client's equity, when the account state is changed. If the maintenance margin is equal to 0, then the initial margin is used.

The symbol should be selected by [Name](#) method.

## MarginLong

Gets the rate of margin charging on long positons.

```
double MarginLong() const
```

### Returned value

Rate of margin charging on long positons.

### Note

The symbol should be selected by [Name](#) method.

## MarginShort

Gets the rate of margin charging on short positons.

```
double MarginShort() const
```

### Returned value

Rate of margin charging on short positons.

### Note

The symbol should be selected by [Name](#) method.

## MarginLimit

Gets the rate of margin charging on Limit orders.

```
double MarginLimit() const
```

### Returned value

Rate of margin charging on Limit orders.

### Note

The symbol should be selected by [Name](#) method.

## MarginStop

Gets the rate of margin charging on Stop orders.

```
double MarginStop() const
```

### Returned value

Rate of margin charging on Stop orders.

### Note

The symbol should be selected by [Name](#) method.

## MarginStopLimit

Gets the rate of margin charging on Stop Limit orders.

```
double MarginStopLimit() const
```

### Returned value

Rate of margin charging on Stop Limit orders.

### Note

The symbol should be selected by [Name](#) method.

## TradeTimeFlags

Gets the flags of the order expiration allowed modes.

```
int TradeTimeFlags() const
```

### Returned value

Flags of the order expiration allowed modes.

### Note

The symbol should be selected by [Name](#) method.



## TradeFillFlags

Gets the flags of the order filling allowed modes.

```
int TradeFillFlags() const
```

### Returned value

Flags of the order filling allowed modes.

### Note

The symbol should be selected by [Name](#) method.

## Digits

Gets the number of digits after period.

```
int Digits() const
```

### Returned value

Gets the number of digits after period.

### Note

The symbol should be selected by [Name](#) method.

## Point

Gets the value of one point.

```
double Point () const
```

### Returned value

Value of one point.

### Note

The symbol should be selected by [Name](#) method.

## TickValue

Gets the cost of tick (minimal change of price).

```
double TickValue() const
```

### Returned value

Cost of tick (minimal change of price).

### Note

The symbol should be selected by [Name](#) method.

## TickValueProfit

Gets the calculated tick price for a profitable position.

```
double TickValueProfit() const
```

### Returned value

The calculated tick price for a profitable position.

### Note

The symbol should be selected by [Name](#) method.

## TicksBookDepth

Gets the calculated tick price for a losing position.

```
double TicksBookDepth() const
```

### Returned value

The calculated tick price for a losing position.

### Note

The symbol should be selected by [Name](#) method.

## TickSize

Gets the minimal change of price.

```
double TickSize() const
```

### Returned value

Minimal change of price.

### Note

The symbol should be selected by [Name](#) method.

## ContractSize

Gets the amount of trade contract.

```
double ContractSize() const
```

### Returned value

Amount of trade contract.

### Note

The symbol should be selected by [Name](#) method.



## LotsMin

Gets the minimal volume to close a deal.

```
double LotsMin() const
```

### Returned value

Minimal volume to close a deal.

### Note

The symbol should be selected by [Name](#) method.

## LotsMax

Gets the maximal volume to close a deal.

```
double LotsMax() const
```

### Returned value

Maximal volume to close a deal.

### Note

The symbol should be selected by [Name](#) method.

## LotsStep

Gets the minimal step of volume change to close a deal.

```
double LotsStep() const
```

### Returned value

Minimal step of volume change to close a deal.

### Note

The symbol should be selected by [Name](#) method.

## LotsLimit

Gets the maximal allowed volume of opened position and pending orders (direction insensitive) for one symbol.

```
double LotsLimit() const
```

### Returned value

The maximal allowed volume of opened position and pending orders (direction insensitive) for one symbol.

### Note

The symbol should be selected by [Name](#) method.

## SwapLong

Gets the value of long position swap.

```
double SwapLong() const
```

### Returned value

Value of long position swap.

### Note

The symbol should be selected by [Name](#) method.

## SwapShort

Gets the value of short position swap.

```
double SwapShort() const
```

### Returned value

Value of short position swap.

### Note

The symbol should be selected by [Name](#) method.

## CurrencyBase

Gets the name of symbol base currency.

```
string CurrencyBase() const
```

### Returned value

Name of symbol base currency.

### Note

The symbol should be selected by [Name](#) method.

## CurrencyProfit

Gets the profit currency name.

```
string CurrencyProfit() const
```

### Returned value

Profit currency name.

### Note

The symbol should be selected by [Name](#) method.



## CurrencyMargin

Gets the margin currency name.

```
string CurrencyMargin() const
```

### Returned value

Margin currency name.

### Note

The symbol should be selected by [Name](#) method.

## Bank

Gets the name of current quote source.

```
string Bank() const
```

### Returned value

Name of current quote source.

### Note

The symbol should be selected by [Name](#) method.

## Description

Gets the string description of symbol.

```
string Description() const
```

### Returned value

String description of symbol.

### Note

The symbol should be selected by [Name](#) method.

## Path

Gets the path in symbols tree.

```
string Path() const
```

### Returned value

Gets the path in symbols tree.

### Note

The symbol should be selected by [Name](#) method.

## SessionDeals

Gets the number of deals in the current session.

```
long SessionDeals() const
```

### Returned value

Number of deals in the current session.

### Note

The symbol should be selected by [Name](#) method.

## SessionBuyOrders

Gets the number of Buy orders at the moment.

```
long SessionBuyOrders() const
```

### Returned value

Number of Buy orders at the moment.

### Note

The symbol should be selected by [Name](#) method.

## SessionSellOrders

Gets then number of Sell orders at the moment.

```
long SessionSellOrders() const
```

### Returned value

Number of Sell orders at the moment.

### Note

The symbol should be selected by [Name](#) method.

## SessionTurnover

Gets summary turnover of the current session.

```
double SessionTurnover() const
```

### Returned value

Summary turnover of the current session.

### Note

The symbol should be selected by [Name](#) method.



## SessionInterest

Gets the summary open interest of the current session.

```
double SessionInterest() const
```

### Returned value

Summary open interest of the current session.

### Note

The symbol should be selected by [Name](#) method.

## SessionBuyOrdersVolume

Gets the current volume of Buy orders.

```
double SessionBuyOrdersVolume () const
```

### Returned value

Current volume of Buy orders.

### Note

The symbol should be selected by [Name](#) method.

## SessionSellOrdersVolume

Gets the current volume of Sell orders.

```
double SessionSellOrdersVolume() const
```

### Returned value

Current volume of Sell orders.

### Note

The symbol should be selected by [Name](#) method.

## SessionOpen

Gets the open price of the current session.

```
double SessionOpen() const
```

### Returned value

Open price of the current session.

### Note

The symbol should be selected by [Name](#) method.

## SessionClose

Gets the close price of the current session.

```
double SessionClose() const
```

### Returned value

Close price of the current session.

### Note

The symbol should be selected by [Name](#) method.

## SessionAW

Gets the average weighted price of the current session.

```
double SessionAW() const
```

### Returned value

Average weighted price of the current session.

### Note

The symbol should be selected by [Name](#) method.

## SessionPriceSettlement

Gets the settlement price of the current session.

```
double SessionPriceSettlement () const
```

### Returned value

Settlement price of the current session.

### Note

The symbol should be selected by [Name](#) method.

## SessionPriceLimitMin

Gets the minimal price of the current session.

```
double SessionPriceLimitMin() const
```

### Returned value

Minimal price of the current session.

### Note

The symbol should be selected by [Name](#) method.



## SessionPriceLimitMax

Gets the maximal price of the current session.

```
double SessionPriceLimitMax() const
```

### Returned value

Maximal price of the current session.

### Note

The symbol should be selected by [Name](#) method.

## InfoInteger

Gets the value of specified integer type property.

```
bool InfoInteger (
    ENUM_SYMBOL_INFO_INTEGER prop_id,    // property ID
    long& var                            // reference to variable
) const
```

### Parameters

*prop\_id*

[in] ID of integer type property (value of [ENUM\\_SYMBOL\\_INFO\\_INTEGER](#) enumeration).

*var*

[out] Reference to long type variable to place result.

### Returned value

true - in case of success, false - if unable to get property value.

### Note

The symbol should be selected by [Name](#) method.

## InfoDouble

Gets the value of specified double type property.

```
bool InfoDouble(  
    ENUM_SYMBOL_INFO_DOUBLE prop_id,    // property ID  
    double& var                        // reference to variable  
) const
```

### Parameters

*prop\_id*

[in] ID of double type property (value of [ENUM\\_SYMBOL\\_INFO\\_DOUBLE](#) enumeration).

*var*

[out] Reference to double type variable to place result.

### Returned value

true - in case of success, false - if unable to get property value.

### Note

The symbol should be selected by [Name](#) method.

## InfoString

Gets the value of specified string type property.

```
bool InfoString(  
    ENUM_SYMBOL_INFO_STRING prop_id,    // property ID  
    string& var                        // reference to variable  
) const
```

### Parameters

*prop\_id*

[in] ID of text property.

*var*

[out] Reference to string type variable to place result.

### Returned value

true - in case of success, false - if unable to get property value.

### Note

The symbol should be selected by [Name](#) method.

## NormalizePrice

Returns the value of price, normalized using the symbol properties.

```
double NormalizePrice(  
    double price // price  
    ) const
```

### Parameters

*price*

[in] Price.

### Returned value

Normalized price.

### Note

The symbol should be selected by [Name](#) method.

## COrderInfo

COrderInfo is a class for easy access to the pending order properties.

### Description

COrderInfo class provides access to the pending order properties.

### Declaration

```
class COrderInfo : public CObject
```

### Title

```
#include <Trade\OrderInfo.mqh>
```

### Class methods by groups

| Access to integer type properties      |                                                                                |
|----------------------------------------|--------------------------------------------------------------------------------|
| <a href="#">Ticket</a>                 | Gets the ticket of an order, previously selected for access                    |
| <a href="#">TimeSetup</a>              | Gets the time of order placement                                               |
| <a href="#">TimeSetupMsc</a>           | Receives the time of placing an order in milliseconds since 01.01.1970         |
| <a href="#">OrderType</a>              | Gets the order type                                                            |
| <a href="#">OrderTypeDescription</a>   | Gets the order type as a string                                                |
| <a href="#">State</a>                  | Gets the order state                                                           |
| <a href="#">StateDescription</a>       | Gets the order state as a string                                               |
| <a href="#">TimeExpiration</a>         | Gets the time of order expiration                                              |
| <a href="#">TimeDone</a>               | Gets the time of order execution or cancellation                               |
| <a href="#">TimeDoneMsc</a>            | Receives order execution or cancellation time in milliseconds since 01.01.1970 |
| <a href="#">TypeFilling</a>            | Gets the type of order execution by remainder                                  |
| <a href="#">TypeFillingDescription</a> | Gets the type of order execution by remainder as a string                      |
| <a href="#">TypeTime</a>               | Gets the type of order at the time of the expiration                           |
| <a href="#">TypeTimeDescription</a>    | Gets the order type by expiration time as a string                             |
| <a href="#">Magic</a>                  | Gets the ID of expert, that placed the order                                   |
| <a href="#">PositionId</a>             | Gets the ID of position                                                        |

|                                         |                                                                 |
|-----------------------------------------|-----------------------------------------------------------------|
| <b>Access to double type properties</b> |                                                                 |
| <a href="#"><u>VolumeInitial</u></a>    | Gets the initial volume of order                                |
| <a href="#"><u>VolumeCurrent</u></a>    | Gets the unfilled volume of order                               |
| <a href="#"><u>PriceOpen</u></a>        | Gets the order price                                            |
| <a href="#"><u>StopLoss</u></a>         | Gets the order's Stop Loss                                      |
| <a href="#"><u>TakeProfit</u></a>       | Gets the order's Take Profit                                    |
| <a href="#"><u>PriceCurrent</u></a>     | Gets the current price by order symbol                          |
| <a href="#"><u>PriceStopLimit</u></a>   | Gets the price of setting limit order                           |
| <b>Access to text properties</b>        |                                                                 |
| <a href="#"><u>Symbol</u></a>           | Gets the name of order symbol                                   |
| <a href="#"><u>Comment</u></a>          | Gets the order comment                                          |
| <b>Access to MQL5 API functions</b>     |                                                                 |
| <a href="#"><u>InfoInteger</u></a>      | Gets the value of specified integer type property               |
| <a href="#"><u>InfoDouble</u></a>       | Gets the value of specified double type property                |
| <a href="#"><u>InfoString</u></a>       | Gets value of specified string type property                    |
| <b>State</b>                            |                                                                 |
| <a href="#"><u>StoreState</u></a>       | Saves the order parameters                                      |
| <a href="#"><u>CheckState</u></a>       | Checks the current parameters against the saved parameters      |
| <b>Selection</b>                        |                                                                 |
| <a href="#"><u>Select</u></a>           | Selects an order by ticket for further access to its properties |
| <a href="#"><u>SelectByIndex</u></a>    | Selects an order by index for further access to its properties  |

## Ticket

Gets the ticket of an order, previously selected for access using the [Select](#) method.

```
ulong Ticket () const
```

### Returned value

Order ticket if successful, otherwise - [ULONG\\_MAX](#).

### Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.



## TimeSetup

Gets the time of order placement.

```
datetime TimeSetup() const
```

### Returned value

Time of order placement.

### Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## TimeSetupMsc

Receives the time of placing an order for execution in milliseconds since 01.01.1970.

```
ulong TimeSetupMsc() const
```

### Returned value

The time of placing an order for execution in milliseconds since 01.01.1970.

### Note

Order should be preliminarily selected for access using [Select](#) (by ticket) or [SelectByIndex](#) (by index) method.

## OrderType

Gets the order type.

```
ENUM_ORDER_TYPE OrderType()
```

### Returned value

Order type (value of [ENUM\\_ORDER\\_TYPE](#) enumeration).

### Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## TypeDescription

Gets the order type as a string.

```
string TypeDescription() const
```

### Returned value

Order type as a string.

### Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## State

Gets the order state.

```
ENUM_ORDER_STATE State() const
```

### Returned value

Order state (value of [ENUM\\_ORDER\\_STATE](#) enumeration).

### Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## StateDescription

Gets the order state as a string.

```
string StateDescription() const
```

### Returned value

Order state as a string.

### Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## TimeExpiration

Gets the order expiration time.

```
datetime TimeExpiration() const
```

### Returned value

Order expiration time, set on its placement.

### Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## TimeDone

Gets the time of order execution or cancellation.

```
datetime TimeDone() const
```

### Returned value

Time of order execution or cancellation.

### Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.



## TimeDoneMsc

Receives order execution or cancellation time in milliseconds since 01.01.1970.

```
ulong TimeDoneMsc() const
```

### Returned value

Order execution or cancellation time in milliseconds since 01.01.1970.

### Note

Order should be preliminarily selected for access using [Select](#) (by ticket) or [SelectByIndex](#) (by index) method.

## TypeFilling

Gets the order filling type.

```
ENUM_ORDER_TYPE_FILLING TypeFilling() const
```

### Returned value

Order filling type (value of [ENUM\\_ORDER\\_TYPE\\_FILLING](#) enumeration).

### Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## TypeFillingDescription

Gets the order filling type as a string.

```
string TypeFillingDescription() const
```

### Returned value

Order filling type as a string.

### Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## TypeTime

Gets the type of order at the time of the expiration.

```
ENUM_ORDER_TYPE_TIME TypeTime () const
```

### Returned value

Type of order at the time of the expiration.

### Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## TypeTimeDescription

Gets the order type by expiration time as a string.

```
string TypeTimeDescription() const
```

### Returned value

Order type by expiration time as a string.

### Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## Magic

Gets the ID of expert, that placed the order.

```
long Magic() const
```

### Returned value

ID of expert, that placed the order.

### Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## PositionId

Gets the ID of position.

```
long PositionId() const
```

### Returned value

ID of position, in which the order was involved.

### Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## VolumeInitial

Gets the initial volume of order.

```
double VolumeInitial () const
```

### Returned value

Initial volume of order.

### Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.



## VolumeCurrent

Gets the unfilled volume of order.

```
double VolumeCurrent() const
```

### Returned value

Unfilled volume of order.

### Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## PriceOpen

Gets the order price.

```
double PriceOpen() const
```

### Returned value

Price of order placement.

### Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## StopLoss

Gets the order's Stop Loss.

```
double StopLoss() const
```

### Returned value

Order's Stop Loss.

### Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## TakeProfit

Gets the order's Take Profit.

```
double TakeProfit() const
```

### Returned value

Order's Take Profit.

### Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## PriceCurrent

Gets the current price by order symbol.

```
double PriceCurrent() const
```

### Returned value

Current price by order symbol.

### Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## PriceStopLimit

Gets the price of setting limit order.

```
double PriceStopLimit() const
```

### Returned value

Price of setting limit order.

### Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## Symbol

Gets the name of order symbol.

```
string Symbol() const
```

### Returned value

Name of order symbol.

### Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## Comment

Gets the order comment.

```
string Comment() const
```

### Returned value

Order comment.

### Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.



## InfoInteger

Gets the value of specified integer type property.

```
bool  InfoInteger (
    ENUM_ORDER_PROPERTY_INTEGER  prop_id,      // property ID
    long&                                var      // reference to variable
) const
```

### Parameters

*prop\_id*

[in] ID of integer type property (value of [ENUM\\_ORDER\\_PROPERTY\\_INTEGER](#) enumeration).

*var*

[out] Reference to long type variable to place result.

### Returned value

true - in case of success, false - if unable to get property value.

### Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## InfoDouble

Gets the value of specified double type property.

```
bool InfoDouble(  
    ENUM_ORDER_PROPERTY_DOUBLE prop_id,    // property ID  
    double& var                            // reference to variable  
) const
```

### Parameters

*prop\_id*

[in] ID of double type property (value of [ENUM\\_ORDER\\_PROPERTY\\_DOUBLE](#) enumeration).

*var*

[out] Reference to double type variable to place result.

### Returned value

true - in case of success, false - if unable to get property value.

### Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## InfoString

Gets the value of specified string type property.

```
bool InfoString(  
    ENUM_ORDER_PROPERTY_STRING prop_id,    // property ID  
    string& var                            // reference to variable  
) const
```

### Parameters

*prop\_id*

[in] ID of text property.

*var*

[out] Reference to string type variable to place result.

### Returned value

true - in case of success, false - if unable to get property value.

### Note

The order should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## StoreState

Saves the order parameters.

```
void StoreState()
```

### Returned value

None.

## CheckState

Checks the current parameters against the saved parameters.

```
bool CheckState()
```

### Returned value

true - if the order parameters have changed since the last call of the [StoreState\(\)](#) method, otherwise  
- false.

## Select

Selects an order by ticket for further access to its properties.

```
bool Select(  
    ulong ticket // order ticket  
)
```

### Returned value

true - in case of success, false - if unable to select order.

## SelectByIndex

Selects an order by index for further access to its properties.

```
bool SelectByIndex(  
    int index // order index  
)
```

### Returned value

true - in case of success, false - if unable to select order.

## CHistoryOrderInfo

CHistoryOrderInfo is a class for easy access to the history order properties.

### Description

CHistoryOrderInfo class provides easy access to the history order properties.

### Declaration

```
class CHistoryOrderInfo : public CObject
```

### Title

```
#include <Trade\HistoryOrderInfo.mqh>
```

### Class methods by groups

| Access to integer type properties      |                                                                                |
|----------------------------------------|--------------------------------------------------------------------------------|
| <a href="#">TimeSetup</a>              | Gets the time of order placement                                               |
| <a href="#">TimeSetupMsc</a>           | Receives the time of placing an order in milliseconds since 01.01.1970         |
| <a href="#">OrderType</a>              | Gets the order type                                                            |
| <a href="#">OrderTypeDescription</a>   | Gets the order type as a string                                                |
| <a href="#">State</a>                  | Gets the order state                                                           |
| <a href="#">StateDescription</a>       | Gets the order state as a string                                               |
| <a href="#">TimeExpiration</a>         | Gets the time of order expiration                                              |
| <a href="#">TimeDone</a>               | Gets the time of order execution or cancellation                               |
| <a href="#">TimeDoneMsc</a>            | Receives order execution or cancellation time in milliseconds since 01.01.1970 |
| <a href="#">TypeFilling</a>            | Gets the type of order execution by remainder                                  |
| <a href="#">TypeFillingDescription</a> | Gets the type of order execution by remainder as a string                      |
| <a href="#">TypeTime</a>               | Gets the type of order at the time of the expiration                           |
| <a href="#">TypeTimeDescription</a>    | Gets the order type by expiration time as a string                             |
| <a href="#">Magic</a>                  | Gets the ID of expert, that placed the order                                   |
| <a href="#">PositionId</a>             | Gets the ID of position                                                        |
| Access to double type properties       |                                                                                |
| <a href="#">VolumeInitial</a>          | Gets the initial volume of order                                               |



|                                       |                                                   |
|---------------------------------------|---------------------------------------------------|
| <a href="#"><u>VolumeCurrent</u></a>  | Gets the unfilled volume of order                 |
| <a href="#"><u>PriceOpen</u></a>      | Gets the order price                              |
| <a href="#"><u>StopLoss</u></a>       | Gets the order's Stop Loss                        |
| <a href="#"><u>TakeProfit</u></a>     | Gets the order's Take Profit                      |
| <a href="#"><u>PriceCurrent</u></a>   | Gets the current price by order symbol            |
| <a href="#"><u>PriceStopLimit</u></a> | Gets the price of setting limit order             |
| <b>Access to text properties</b>      |                                                   |
| <a href="#"><u>Symbol</u></a>         | Gets the order symbol                             |
| <a href="#"><u>Comment</u></a>        | Gets the order comment                            |
| <b>Access to MQL5 API functions</b>   |                                                   |
| <a href="#"><u>InfoInteger</u></a>    | Gets the value of specified integer type property |
| <a href="#"><u>InfoDouble</u></a>     | Gets the value of specified double type property  |
| <a href="#"><u>InfoString</u></a>     | Gets value of specified string type property      |
| <b>Selection</b>                      |                                                   |
| <a href="#"><u>Ticket</u></a>         | Gets the ticket/selects the order                 |
| <a href="#"><u>SelectByIndex</u></a>  | Selects the order by index                        |

## TimeSetup

Gets the time of order placement.

```
datetime TimeSetup() const
```

### Returned value

Time of order placement.

### Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## TimeSetupMsc

Receives the time of placing an order for execution in milliseconds since 01.01.1970.

```
ulong TimeSetupMsc() const
```

### Returned value

The time of placing an order for execution in milliseconds since 01.01.1970.

### Note

Historical order should be preliminarily selected for access using [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) method.

## OrderType

Gets the order type.

```
ENUM_ORDER_TYPE OrderType() const
```

### Returned value

Order type (value of [ENUM\\_ORDER\\_TYPE](#) enumeration).

### Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## TypeDescription

Gets the order type as a string.

```
string TypeDescription() const
```

### Returned value

Order type as a string.

### Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## State

Gets the order state.

```
ENUM_ORDER_STATE State() const
```

### Returned value

Order state (value of [ENUM\\_ORDER\\_STATE](#) enumeration).

### Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## StateDescription

Gets the order state as a string.

```
string StateDescription() const
```

### Returned value

Order state as a string.

### Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## TimeExpiration

Gets the time of order expiration.

```
datetime TimeExpiration() const
```

### Returned value

Time of order expiration, set on its placement.

### Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.



## TimeDone

Gets the time of order execution or cancellation.

```
datetime TimeDone() const
```

### Returned value

Time of order execution or cancellation.

### Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## TimeDoneMsc

Receives order execution or cancellation time in milliseconds since 01.01.1970.

```
ulong TimeDoneMsc() const
```

### Returned value

Order execution or cancellation time in milliseconds since 01.01.1970.

### Note

Historical order should be preliminarily selected for access using [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) method.

## TypeFilling

Gets the type of order execution by remainder.

```
ENUM_ORDER_TYPE_FILLING TypeFilling() const
```

### Returned value

Type of order execution by remainder (value of [ENUM\\_ORDER\\_TYPE\\_FILLING](#) enumeration).

### Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## TypeFillingDescription

Gets the type of order execution by remainder as a string.

```
string TypeFillingDescription() const
```

### Returned value

Type order of execution by remainder as a string.

### Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## TypeTime

Gets the type of order at the time of the expiration.

```
ENUM_ORDER_TYPE_TIME TypeTime () const
```

### Returned value

Type of order at the time of the expiration (value of [ENUM\\_ORDER\\_TYPE\\_TIME](#) enumeration).

### Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## TypeTimeDescription

Gets the order type by expiration time as a string.

```
string TypeTimeDescription() const
```

### Returned value

Order type by expiration time as a string.

### Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## Magic

Gets the ID of the Expert Advisor, that placed the order.

```
long Magic() const
```

### Returned value

ID of the Expert Advisor, that placed the order.

### Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## PositionId

Gets the ID of position.

```
long PositionId() const
```

### Returned value

ID of position, in which the order was involved.

### Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.



## VolumeInitial

Gets the initial volume of order.

```
double VolumeInitial() const
```

### Returned value

Initial volume of order.

### Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## VolumeCurrent

Gets the unfilled volume of order.

```
double VolumeCurrent() const
```

### Returned value

Unfilled volume of order.

### Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## PriceOpen

Gets the order price.

```
double PriceOpen() const
```

### Returned value

Price of order placement.

### Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## StopLoss

Gets the Stop Loss price of the order.

```
double StopLoss() const
```

### Returned value

Stop Loss price of the order.

### Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## TakeProfit

Gets the Take Profit price of the order.

```
double TakeProfit() const
```

### Returned value

The Take Profit price of the order.

### Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## PriceCurrent

Gets the current price of the order's symbol.

```
double PriceCurrent() const
```

### Returned value

The current price of order's symbol.

### Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## PriceStopLimit

Gets the stop limit price of the order.

```
double PriceStopLimit() const
```

### Returned value

Stop Limit price of the order.

### Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## Symbol

Gets the name of order symbol.

```
string Symbol() const
```

### Returned value

Name of order symbol.

### Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.



## Comment

Gets the order comment.

```
string Comment() const
```

### Returned value

Order comment.

### Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## InfoInteger

Gets the value of specified integer type property.

```
bool InfoInteger (
    ENUM_ORDER_PROPERTY_INTEGER prop_id,    // property ID
    long& var                                // reference to variable
) const
```

### Parameters

*prop\_id*

[in] ID of integer type property (value of [ENUM\\_ORDER\\_PROPERTY\\_INTEGER](#) enumeration).

*var*

[out] Reference to long type variable to place result.

### Returned value

true - in case of success, false - if unable to get property value.

### Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## InfoDouble

Gets the value of specified double type property.

```
bool InfoDouble(  
    ENUM_ORDER_PROPERTY_DOUBLE prop_id,    // property ID  
    double& var                            // reference to variable  
) const
```

### Parameters

*prop\_id*

[in] ID of double type property (value of [ENUM\\_ORDER\\_PROPERTY\\_DOUBLE](#) enumeration).

*var*

[out] Reference to double type variable to place result.

### Returned value

true - in case of success, false - if unable to get property value.

### Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## InfoString

Gets the value of specified string type property.

```
bool InfoString(  
    ENUM_ORDER_PROPERTY_STRING prop_id,    // property ID  
    string& var                            // reference to variable  
) const
```

### Parameters

*prop\_id*

[in] ID of text property (value of [ENUM\\_ORDER\\_PROPERTY\\_STRING](#) enumeration).

*var*

[out] Reference to string type variable to place result.

### Returned value

true - in case of success, false - if unable to get property value.

### Note

The historical order should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## Ticket (Get method)

Gets the order ticket.

```
ulong Ticket() const
```

### Returned value

Order ticket.

## Ticket (Set method)

Select the order for further work.

```
void Ticket(  
    ulong ticket    // order ticket  
)
```

### Parameters

*ticket*

[in] Order ticket.

## SelectByIndex

Selects an order by index for further access to its properties.

```
bool SelectByIndex(  
    int index // order index  
)
```

### Returned value

true - in case of success, false - if unable to select order.

## CPositionInfo

CPositionInfo is a class for easy access to the open position properties.

### Description

CPositionInfo class provides easy access to the open position properties.

### Declaration

```
class CPositionInfo : public CObject
```

### Title

```
#include <Trade\PositionInfo.mqh>
```

### Class methods by groups

|                                          |                                                                         |
|------------------------------------------|-------------------------------------------------------------------------|
| <b>Access to integer type properties</b> |                                                                         |
| <a href="#"><u>Time</u></a>              | Gets the time of position opening                                       |
| <a href="#"><u>TimeMsc</u></a>           | Receives the time of position opening in milliseconds since 01.01.1970  |
| <a href="#"><u>TimeUpdate</u></a>        | Receives the time of position changing in seconds since 01.01.1970      |
| <a href="#"><u>TimeUpdateMsc</u></a>     | Receives the time of position changing in milliseconds since 01.01.1970 |
| <a href="#"><u>PositionType</u></a>      | Gets the position type                                                  |
| <a href="#"><u>TypeDescription</u></a>   | Gets the position type as a string                                      |
| <a href="#"><u>Magic</u></a>             | Gets the ID of expert, that opened the position                         |
| <a href="#"><u>Identifier</u></a>        | Gets the ID of position                                                 |
| <b>Access to double type properties</b>  |                                                                         |
| <a href="#"><u>Volume</u></a>            | Gets the volume of position                                             |
| <a href="#"><u>PriceOpen</u></a>         | Gets the price of position opening                                      |
| <a href="#"><u>StopLoss</u></a>          | Gets the price of position's Stop Loss                                  |
| <a href="#"><u>TakeProfit</u></a>        | Gets the price of position's Take Profit                                |
| <a href="#"><u>PriceCurrent</u></a>      | Gets the current price by position symbol                               |
| <a href="#"><u>Commission</u></a>        | Gets the amount of commission by position                               |
| <a href="#"><u>Swap</u></a>              | Gets the amount of swap by position                                     |
| <a href="#"><u>Profit</u></a>            | Gets the amount of current profit by position                           |
| <b>Access to text properties</b>         |                                                                         |

|                                     |                                                            |
|-------------------------------------|------------------------------------------------------------|
| <a href="#">Symbol</a>              | Gets the name of position symbol                           |
| <a href="#">Comment</a>             | Gets the comment of the position                           |
| <b>Access to MQL5 API functions</b> |                                                            |
| <a href="#">InfoInteger</a>         | Gets the value of specified integer type property          |
| <a href="#">InfoDouble</a>          | Gets the value of specified double type property           |
| <a href="#">InfoString</a>          | Gets the value of specified string type property           |
| <b>Selection</b>                    |                                                            |
| <a href="#">Select</a>              | Selects the position                                       |
| <a href="#">SelectByIndex</a>       | Selects the position by index                              |
| <b>State</b>                        |                                                            |
| <a href="#">StoreState</a>          | Saves the position parameters                              |
| <a href="#">CheckState</a>          | Checks the current parameters against the saved parameters |



## Time

Gets the time of position opening.

```
datetime Time() const
```

### Returned value

Time of position opening.

### Note

The position should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## TimeMsc

Receives position opening time in milliseconds since 01.01.1970.

```
ulong TimeMsc() const
```

### Returned value

Position opening time in milliseconds since 01.01.1970.

### Note

Position should be preliminarily selected for access using [Select](#) (by symbol) or [SelectByIndex](#) (by index) method.

## TimeUpdate

Receives the time of position changing in seconds since 01.01.1970.

```
datetime TimeUpdate() const
```

### Returned value

Time of position changing in seconds since 01.01.1970.

### Note

Position should be preliminarily selected for access using [Select](#) (by symbol) or [SelectByIndex](#) (by index) method.

## TimeUpdateMsc

Receives the time of position changing in milliseconds since 01.01.1970.

```
ulong TimeUpdateMsc() const
```

### Returned value

The time of position changing in milliseconds since 01.01.1970.

### Note

Position should be preliminarily selected for access using [Select](#) (by symbol) or [SelectByIndex](#) (by index) method.

## PositionType

Gets the position type.

```
ENUM_POSITION_TYPE PositionType() const
```

### Returned value

Position type (value of [ENUM\\_POSITION\\_TYPE](#) enumeration).

### Note

The position should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## TypeDescription

Gets the position type as a string.

```
string TypeDescription() const
```

### Returned value

Position type as a string.

### Note

The position should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## Magic

Gets the ID of Expert Advisor, opened the position.

```
long Magic() const
```

### Returned value

ID of the Expert Advisor, opened the position.

### Note

The position should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## Identifier

Gets the ID of position.

```
long Identifier() const
```

### Returned value

ID of position.

### Note

The position should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.



## Volume

Gets the volume of position.

```
double Volume() const
```

### Returned value

Volume of position.

### Note

The position should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## PriceOpen

Gets the price of position opening.

```
double PriceOpen() const
```

### Returned value

Position open price.

### Note

The position should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## StopLoss

Gets the Stop Loss price of the position.

```
double StopLoss() const
```

### Returned value

The Stop Loss price of the position.

### Note

The position should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## TakeProfit

Gets the Take Profit price of the position.

```
double TakeProfit() const
```

### Returned value

The Take Profit price of the position.

### Note

The position should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## PriceCurrent

Gets the current price by position symbol.

```
double PriceCurrent() const
```

### Returned value

Current price by position symbol.

## Commission

Gets the amount of commission of the position.

```
double Commission() const
```

### Returned value

Amount of commission of the position (in deposit currency).

### Note

The position should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## Swap

Gets the amount of swap of the position.

```
double Swap() const
```

### Returned value

Amount of swap of the position (in deposit currency).

### Note

The position should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## Profit

Gets the amount of current profit of the position.

```
double Profit() const
```

### Returned value

Amount of current profit of the position (in deposit currency).

### Note

The position should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.



## Symbol

Gets the name of position symbol.

```
string Symbol() const
```

### Returned value

Name of position symbol.

### Note

The position should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## Comment

Gets the comment of the position.

```
string Comment() const
```

### Returned value

Comment of the position.

### Note

The position should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## InfoInteger

Gets the value of specified integer type property.

```
bool InfoInteger(  
    ENUM_POSITION_PROPERTY_INTEGER prop_id,    // property ID  
    long& var                                // reference to variable  
) const
```

### Parameters

*prop\_id*

[in] ID of integer type property (value of [ENUM\\_POSITION\\_PROPERTY\\_INTEGER](#) enumeration).

*var*

[out] Reference to long type variable to place result.

### Returned value

true - in case of success, false - if unable to get property value.

### Note

The position should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## InfoDouble

Gets the value of specified double type property.

```
bool InfoDouble(  
    ENUM_POSITION_PROPERTY_DOUBLE prop_id,    // property ID  
    double& var                               // reference to variable  
) const
```

### Parameters

*prop\_id*

[in] ID of double type property (value of [ENUM\\_POSITION\\_PROPERTY\\_DOUBLE](#) enumeration).

*var*

[in] Reference to double type variable to place result.

### Returned value

true - in case of success, false - if unable to get property value.

### Note

The position should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## InfoString

Gets the value of specified string type property.

```
bool InfoString(  
    ENUM_POSITION_PROPERTY_STRING prop_id,    // property ID  
    string& var                               // reference to variable  
) const
```

### Parameters

*prop\_id*

[in] ID of text property (value of [ENUM\\_POSITION\\_PROPERTY\\_STRING](#) enumeration).

*var*

[out] Reference to string type variable to place result.

### Returned value

true - in case of success, false - if unable to get property value.

### Note

The position should be selected using the [Select](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## Select

Select the position for further work.

```
bool Select(  
    const string symbol    // symbol  
)
```

### Parameters

*symbol*

[in] Symbol for position selection.

## SelectByIndex

Selects the position by index for further access to its properties.

```
bool SelectByIndex(  
    int index // position index  
)
```

### Returned value

true - in case of success, false - if unable to select position.

## StoreState

Saves the position parameters.

```
void StoreState()
```

### Returned value

None.



## CheckState

Checks the current parameters against the saved parameters.

```
bool CheckState()
```

### Returned value

true - if the position parameters have changed since the last call of the [StoreState\(\)](#) method,  
otherwise - false.

## CDealInfo

CDealInfo is a class for easy access to the deal properties.

### Description

CDealInfo class provides access to the deal properties.

### Declaration

```
class CDealInfo : public CObject
```

### Title

```
#include <Trade\DealInfo.mqh>
```

### Class methods by groups

|                                   |                                                                        |
|-----------------------------------|------------------------------------------------------------------------|
| Access to integer type properties |                                                                        |
| <a href="#">Order</a>             | Gets the order by which the deal is executed                           |
| <a href="#">Time</a>              | Gets the time of deal execution                                        |
| <a href="#">TimeMsc</a>           | Receives the time of a deal execution in milliseconds since 01.01.1970 |
| <a href="#">DealType</a>          | Gets the deal type                                                     |
| <a href="#">TypeDescription</a>   | Gets the deal type as a string                                         |
| <a href="#">Entry</a>             | Gets the deal direction                                                |
| <a href="#">EntryDescription</a>  | Gets the deal direction as a string                                    |
| <a href="#">Magic</a>             | Gets the ID of expert, that executed the deal                          |
| <a href="#">PositionId</a>        | Gets the ID of position, in which the deal was involved                |
| Access to double type properties  |                                                                        |
| <a href="#">Volume</a>            | Gets the volume of deal                                                |
| <a href="#">Price</a>             | Gets the deal price                                                    |
| <a href="#">Commision</a>         | Gets the amount of commission by deal                                  |
| <a href="#">Swap</a>              | Gets the amount of swap when position is closed                        |
| <a href="#">Profit</a>            | Gets the financial result of deal                                      |
| Access to text properties         |                                                                        |
| <a href="#">Symbol</a>            | Gets the name of deal symbol                                           |
| <a href="#">Comment</a>           | Gets the deal comment                                                  |

|                               |                                                   |
|-------------------------------|---------------------------------------------------|
| Access to MQL5 API functions  |                                                   |
| <a href="#">InfoInteger</a>   | Gets the value of specified integer type property |
| <a href="#">InfoDouble</a>    | Gets the value of specified double type property  |
| <a href="#">InfoString</a>    | Gets value of specified string type property      |
| Selection                     |                                                   |
| <a href="#">Ticket</a>        | Gets ticket/selects the deal                      |
| <a href="#">SelectByIndex</a> | Selects the deal by index                         |

## Order

Gets the order by which the deal is executed.

```
long Order() const
```

### Returned value

Order by which the deal is executed.

### Note

The deal should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## Time

Gets the time of deal execution.

```
datetime Time() const
```

### Returned value

Time of deal execution.

### Note

The deal should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## TimeMsc

Receives the time of a deal execution in milliseconds since 01.01.1970.

```
ulong TimeMsc() const
```

### Returned value

The time of a deal execution in milliseconds since 01.01.1970.

### Note

Deal should be preliminarily selected for access using [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) method.

## DealType

Gets the deal type.

```
ENUM_DEAL_TYPE DealType() const
```

### Returned value

Deal type (value of [ENUM\\_DEAL\\_TYPE](#) enumeration).

### Note

The deal should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## TypeDescription

Gets the deal type as a string.

```
string TypeDescription() const
```

### Returned value

Deal type as a string.

### Note

The deal should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.



## Entry

Gets the deal direction.

```
ENUM_DEAL_ENTRY Entry() const
```

### Returned value

Deal direction (value of [ENUM\\_DEAL\\_ENTRY](#) enumeration.).

### Note

The deal should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## EntryDescription

Gets the deal direction as a string.

```
string EntryDescription() const
```

### Returned value

Deal direction as a string.

### Note

The deal should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## Magic

Gets the ID of the Expert Advisor, that executed the deal.

```
long Magic() const
```

### Returned value

ID of the Expert Advisor, that executed the deal.

### Note

The deal should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## PositionId

Gets the ID of position, in which the deal was involved.

```
long PositionId() const
```

### Returned value

ID of position, in which the deal was involved.

### Note

The deal should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## Volume

Gets the volume of deal.

```
double Volume() const
```

### Returned value

Volume of deal.

### Note

The deal should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## Price

Gets the deal price.

```
double Price() const
```

### Returned value

Deal price.

### Note

The deal should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## Commission

Gets the amount of commission of the deal.

```
double Commission() const
```

### Returned value

Amount of commission of the deal.

### Note

The deal should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## Swap

Gets the amount of swap when position is closed.

```
double Swap() const
```

### Returned value

Amount of swap when position is closed.

### Note

The deal should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.



## Profit

Gets the financial result of the deal.

```
double Profit() const
```

### Returned value

Financial result of the deal (in deposit currency).

### Note

The deal should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## Symbol

Gets the name of the deal symbol.

```
string Symbol() const
```

### Returned value

Name of the deal symbol.

### Note

The deal should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## Comment

Gets the deal comment.

```
string Comment() const
```

### Returned value

Deal comment.

### Note

The deal should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## InfoInteger

Gets the value of specified integer type property.

```
bool InfoInteger (
    ENUM_DEAL_PROPERTY_INTEGER prop_id,    // property ID
    long& var                               // reference to variable
) const
```

### Parameters

*prop\_id*

[in] ID of integer type property (value of [ENUM\\_DEAL\\_PROPERTY\\_INTEGER](#) enumeration).

*var*

[out] Reference to long type variable to place result.

### Returned value

true - in case of success, false - if unable to get property value.

### Note

The deal should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## InfoDouble

Gets the value of specified double type property.

```
bool InfoDouble(  
    ENUM_DEAL_PROPERTY_DOUBLE prop_id,    // property ID  
    double& var                          // reference to variable  
) const
```

### Parameters

*prop\_id*

[in] ID of double type property (value of [ENUM\\_DEAL\\_PROPERTY\\_DOUBLE](#) enumeration).

*var*

[in] Reference to double type variable to place result.

### Returned value

true - in case of success, false - if unable to get property value.

### Note

The deal should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## InfoString

Gets the value of specified string type property.

```
bool InfoString(  
    ENUM_DEAL_PROPERTY_STRING prop_id,    // property ID  
    string& var                            // reference to variable  
) const
```

### Parameters

*prop\_id*

[in] ID of text property (value of [ENUM\\_DEAL\\_PROPERTY\\_STRING](#) enumeration).

*var*

[out] Reference to string type variable to place result.

### Returned value

true - in case of success, false - if unable to get property value.

### Note

The deal should be selected using the [Ticket](#) (by ticket) or [SelectByIndex](#) (by index) methods.

## Ticket (Get method)

Gets the deal ticket.

```
ulong Ticket() const
```

### Returned value

Deal ticket.

## Ticket (Set method)

Select the position for further work.

```
void Ticket(  
    ulong ticket    // ticket  
)
```

### Parameters

*ticket*

[in] Deal ticket.

## SelectByIndex

Selects the deal by index for further access to its properties.

```
bool SelectByIndex(  
    int index // order index  
)
```

### Returned value

true - in case of success, false - if unable to select the deal.



## CTrade

CTrade is a class for easy access to the trade functions.

### Description

CTrade class provides easy access to the trade functions.

### Declaration

```
class CTrade : public CObject
```

### Title

```
#include <Trade\Trade.mqh>
```

### Class methods by groups

| Setting parameters                   |                                                                      |
|--------------------------------------|----------------------------------------------------------------------|
| <a href="#">LogLevel</a>             | Sets logging level                                                   |
| <a href="#">SetExpertMagicNumber</a> | Sets the expert ID                                                   |
| <a href="#">SetDeviationInPoints</a> | Sets the allowed deviation                                           |
| <a href="#">SetTypeFilling</a>       | Sets filling type of the order                                       |
| <a href="#">SetAsyncMode</a>         | Sets asynchronous mode for trade operations                          |
| Operations with orders               |                                                                      |
| <a href="#">OrderOpen</a>            | Places the pending order with set parameters                         |
| <a href="#">OrderModify</a>          | Modifies the pending order parameters                                |
| <a href="#">OrderDelete</a>          | Deletes the pending order                                            |
| Operations with positions            |                                                                      |
| <a href="#">PositionOpen</a>         | Opens the position with set parameters                               |
| <a href="#">PositionModify</a>       | Modifies the position parameters                                     |
| <a href="#">PositionClose</a>        | Closes the position                                                  |
| Additional methods                   |                                                                      |
| <a href="#">Buy</a>                  | Opens a long position with specified parameters                      |
| <a href="#">Sell</a>                 | Opens a short position with specified parameters                     |
| <a href="#">BuyLimit</a>             | Places the pending order of Buy Limit type with specified parameters |
| <a href="#">BuyStop</a>              | Places the pending order of Buy Stop type with specified parameters  |

|                                                      |                                                                              |
|------------------------------------------------------|------------------------------------------------------------------------------|
| <a href="#"><u>SellLimit</u></a>                     | Places the pending order of Sell Limit type with specified parameters        |
| <a href="#"><u>SellStop</u></a>                      | Places the pending order of Sell Stop type with specified parameters         |
| <b>Access to the last request parameters</b>         |                                                                              |
| <a href="#"><u>Request</u></a>                       | Gets the copy of the last request structure                                  |
| <a href="#"><u>RequestAction</u></a>                 | Gets the trade operation type                                                |
| <a href="#"><u>RequestActionDescription</u></a>      | Gets the trade operation type as string                                      |
| <a href="#"><u>RequestMagic</u></a>                  | Gets the magic number of the Expert Advisor                                  |
| <a href="#"><u>RequestOrder</u></a>                  | Gets the order ticket, used in the last request                              |
| <a href="#"><u>RequestSymbol</u></a>                 | Gets the name of the symbol, used in the last request                        |
| <a href="#"><u>RequestVolume</u></a>                 | Gets the trade volume (in lots), used in the last request                    |
| <a href="#"><u>RequestPrice</u></a>                  | Gets the price, used in the last request                                     |
| <a href="#"><u>RequestStopLimit</u></a>              | Gets the price of pending order of Stop Limit type, used in the last request |
| <a href="#"><u>RequestSL</u></a>                     | Gets the Stop Loss price of the order, used in the last request              |
| <a href="#"><u>RequestTP</u></a>                     | Gets the Take Profit price of the order, used in the last request            |
| <a href="#"><u>RequestDeviation</u></a>              | Gets the price deviation of the order, used in the last request              |
| <a href="#"><u>RequestType</u></a>                   | Gets the type of the order, used in the last request                         |
| <a href="#"><u>RequestTypeDescription</u></a>        | Gets the type of the order (as string) , used in the last request            |
| <a href="#"><u>RequestTypeFilling</u></a>            | Gets the filling type of the order, used in the last request                 |
| <a href="#"><u>RequestTypeFillingDescription</u></a> | Gets the filling type of the order (as string), used in the last request     |
| <a href="#"><u>RequestTypeTime</u></a>               | Gets the validity period of the order, used in the last request              |
| <a href="#"><u>RequestTypeTimeDescription</u></a>    | Gets the validity period of the order (as string), used in the last request  |
| <a href="#"><u>RequestExpiration</u></a>             | Gets the expiration time of the order, used in the last request              |
| <a href="#"><u>RequestComment</u></a>                | Gets the comment of the order, used in the last                              |

|                                                      |                                                                                                                                                       |
|------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                      | request                                                                                                                                               |
| <b>Access to the last request checking results</b>   |                                                                                                                                                       |
| <a href="#"><u>CheckResult</u></a>                   | Gets the copy of the structure of the last request check result.                                                                                      |
| <a href="#"><u>CheckResultRetcode</u></a>            | Gets the value of the retcode field of <a href="#"><u>MqlTradeCheckResult</u></a> type, filled while checking of the request correctness              |
| <a href="#"><u>CheckResultRetcodeDescription</u></a> | Gets the string description of the retcode field of <a href="#"><u>MqlTradeCheckResult</u></a> type, filled while checking of the request correctness |
| <a href="#"><u>CheckResultBalance</u></a>            | Gets the value of the balance field of <a href="#"><u>MqlTradeCheckResult</u></a> type, filled while checking of the request correctness              |
| <a href="#"><u>CheckResultEquity</u></a>             | Gets the value of the equity field of <a href="#"><u>MqlTradeCheckResult</u></a> type, filled while checking of the request correctness               |
| <a href="#"><u>CheckResultProfit</u></a>             | Gets the value of the profit field of <a href="#"><u>MqlTradeCheckResult</u></a> type, filled while checking of the request correctness               |
| <a href="#"><u>CheckResultMargin</u></a>             | Gets the value of the margin field of <a href="#"><u>MqlTradeCheckResult</u></a> type, filled while checking of the request correctness               |
| <a href="#"><u>CheckResultMarginFree</u></a>         | Gets the value of the margin_free field of <a href="#"><u>MqlTradeCheckResult</u></a> type, filled while checking of the request correctness          |
| <a href="#"><u>CheckResultMarginLevel</u></a>        | Gets the value of the margin_level field of <a href="#"><u>MqlTradeCheckResult</u></a> type, filled while checking of the request correctness         |
| <a href="#"><u>CheckResultComment</u></a>            | Gets the value of the comment field of <a href="#"><u>MqlTradeCheckResult</u></a> type, filled while checking of the request correctness              |
| <b>Access to the last request execution results</b>  |                                                                                                                                                       |
| <a href="#"><u>Result</u></a>                        | Gets the copy of the structure of the last request result                                                                                             |
| <a href="#"><u>ResultRetcode</u></a>                 | Gets the code of request result                                                                                                                       |
| <a href="#"><u>ResultRetcodeDescription</u></a>      | Gets the code of request result as text                                                                                                               |
| <a href="#"><u>ResultDeal</u></a>                    | Gets the deal ticket                                                                                                                                  |
| <a href="#"><u>ResultOrder</u></a>                   | Gets the order ticket                                                                                                                                 |
| <a href="#"><u>ResultVolume</u></a>                  | Gets the volume of deal or order                                                                                                                      |
| <a href="#"><u>ResultPrice</u></a>                   | Gets the price, confirmed by broker                                                                                                                   |

|                                            |                                                                          |
|--------------------------------------------|--------------------------------------------------------------------------|
| <a href="#"><u>ResultBid</u></a>           | Gets the current bid price (the requote)                                 |
| <a href="#"><u>ResultAsk</u></a>           | Gets the current ask price (the requote)                                 |
| <a href="#"><u>ResultComment</u></a>       | Gets the broker comment                                                  |
| <b>Auxiliary methods</b>                   |                                                                          |
| <a href="#"><u>PrintRequest</u></a>        | Prints the last request parameters into journal                          |
| <a href="#"><u>PrintResult</u></a>         | Prints the results of the last request into journal                      |
| <a href="#"><u>FormatRequest</u></a>       | Prepares the formatted string with last request parameters               |
| <a href="#"><u>FormatRequestResult</u></a> | Prepares the formatted string with results of the last request execution |

# LogLevel

Sets logging level for messages.

```
void LogLevel(  
    ENUM_LOG_LEVELS log_level // logging level  
)
```

### Parameters

*log\_level*  
[in] Logging level.

### Returned value

None.

### Note

LOG\_LEVEL\_NO and less disables displaying any messages (set up automatically in the optimization mode). LOG\_LEVEL\_ERRORS enables displaying only error messages (value by default). LOG\_LEVEL\_ALL and greater enables displaying any messages (set up automatically in the test mode).

### ENUM\_LOG\_LEVELS

| Идентификатор    | Описание                          | Значение |
|------------------|-----------------------------------|----------|
| LOG_LEVEL_NO     | Displaying messages disabled      | 0        |
| LOG_LEVEL_ERRORS | Only error messages are displayed | 1        |
| LOG_LEVEL_ALL    | All messages are displayed        | 2        |

## SetExpertMagicNumber

Sets the expert ID.

```
void SetExpertMagicNumber (
    ulong magic    // ID
)
```

### Parameters

*magic*

[in] New ID of the expert.

### Returned value

None.

## SetDeviationInPoints

Sets the allowed deviation.

```
void SetDeviationInPoints(  
    ulong deviation    // deviation  
)
```

### Parameters

*deviation*

[in] Allowed deviation.

### Returned value

None.

## SetTypeFilling

Sets filling type of the order.

```
void SetTypeFilling(  
    ENUM_ORDER_TYPE_FILLING filling    // order filling type  
)
```

### Parameters

*filling*

[in] Order filling type (value of [ENUM\\_ORDER\\_TYPE\\_FILLING](#) enumeration).

### Returned value

None.



## SetAsyncMode

Sets asynchronous mode for trade operations.

```
void SetAsyncMode(  
    bool mode    // asynchronous mode flag  
)
```

### Parameters

*mode*

[in] Asynchronous mode flag.

### Returned value

None.

### Note

This mode is used for asynchronous (without waiting for the trade server's response to a sent request) trade operations (see [OrderSendAsync](#)).

## OrderOpen

Places the pending order with set parameters.

```
bool OrderOpen(
    const string      symbol,           // symbol
    ENUM_ORDER_TYPE   order_type,      // order type
    double            volume,          // order volume
    double            limit_price,      // StopLimit price
    double            price,           // execution price
    double            sl,              // Stop Loss price
    double            tp,              // Take Profit price
    ENUM_ORDER_TYPE_TIME type_time,     // type by expiration
    datetime          expiration,      // expiration
    const string      comment=""       // comment
)
```

### Parameters

*symbol*

[in] Name of trade instrument.

*order\_type*

[in] Type of order trade operation (value of [ENUM\\_ORDER\\_TYPE](#) enumeration).

*volume*

[in] Requested order volume.

*limit\_price*

[in] Price at which the StopLimit order will be placed.

*price*

[in] Price at which the order must be executed.

*sl*

[in] Price at which the Stop Loss will trigger.

*tp*

[in] Price at which the Take Profit will trigger.

*type\_time*

[in] Order type by execution (value of [ENUM\\_ORDER\\_TYPE\\_TIME](#) enumeration).

*expiration*

[in] Expiration date of pending order.

*comment=""*

[in] Order comment.

### Returned value

true - in case of successful check of the basic structures, otherwise - false.

**Note**

Successful completion of the `OrderSend(...)` method does not always mean successful execution of the trade operation. It's necessary to check the result of trade request (trade server return code) using [ResultRetcode\(\)](#) and value, returned by [ResultOrder\(\)](#).

## OrderModify

Modifies the pending order parameters.

```
bool OrderModify(  
    ulong          ticket,          // order ticket  
    double         price,           // execution price  
    double         sl,              // Stop Loss price  
    double         tp,              // Take Profit price  
    ENUM_ORDER_TYPE_TIME type_time, // type by expiration  
    datetime       expiration,      // expiration  
    double         stoplimit        // Limit order price  
)
```

### Parameters

*ticket*

[in] Order ticket.

*price*

[in] The new price by which the order must be executed (or the previous value, if the change is not necessary).

*sl*

[in] The new price by which the Stop Loss will trigger (or the previous value, if the change is not necessary).

*tp*

[in] The new price by which the Take Profit will trigger (or the previous value, if the change is not necessary).

*type\_time*

[in] The new type of order by expiration (or the previous value, if the change is not necessary), value of [ENUM\\_ORDER\\_TYPE\\_TIME](#) enumeration.

*expiration*

[in] The new expiration date of pending order (or the previous value, if the change is not necessary).

*stoplimit*

[in] New price used for setting a Limit order when the price reaches *price* value. It is specified only for StopLimit orders.

### Returned value

true - in case of successful check of the basic structures, otherwise - false.

### Note

Successful completion of the OrderModify(...) method does not always mean successful execution of the trade operation. It's necessary to check the result of trade request (trade server return code) using [ResultRetcode\(\)](#).

## OrderDelete

Deletes the pending order.

```
bool OrderDelete(  
    ulong ticket    // order ticket  
)
```

### Parameters

*ticket*

[in] Order ticket.

### Returned value

true - in case of successful check of the basic structures, otherwise - false.

### Note

Successful completion of the OrderDelete(...) method does not always mean successful execution of the trade operation. It's necessary to check the result of trade request (trade server return code) using [ResultRetcode\(\)](#).

## PositionOpen

Opens the position with set parameters.

```
bool PositionOpen(  
    const string      symbol,           // symbol  
    ENUM_ORDER_TYPE   order_type,      // order type to open position  
    double            volume,          // position volume  
    double            price,           // execution price  
    double            sl,              // Stop Loss price  
    double            tp,              // Take Profit price  
    const string      comment=""       // comment  
)
```

### Parameters

*symbol*

[in] Name of trade instrument, by which it is intended to open position.

*order\_type*

[in] Order type (trade operation) to open position (value of [ENUM\\_ORDER\\_TYPE](#) enumeration).

*volume*

[in] Requested position volume.

*price*

[in] Price at which the position must be opened.

*sl*

[in] Price at which the Stop Loss will trigger.

*tp*

[in] Price at which the Take Profit will trigger.

*comment=""*

[in] Position comment.

### Returned value

true - in case of successful check of the basic structures, otherwise - false.

### Note

Successful completion of the `PositionOpen(...)` method does not always mean successful execution of the trade operation. It's necessary to check the result of trade request (trade server return code) using [ResultRetcode\(\)](#) and value, returned by [ResultDeal\(\)](#).

## PositionModify

Modifies the position parameters by specified symbol.

```
bool PositionModify(  
    const string symbol,      // symbol  
    double      sl,          // Stop Loss price  
    double      tp           // Take Profit price  
)
```

### Parameters

*symbol*

[in] Name of trade instrument, by which it is intended to modify position.

*sl*

[in] The new price by which the Stop Loss will trigger (or the previous value, if the change is not necessary).

*tp*

[in] The new price by which the Take Profit will trigger (or the previous value, if the change is not necessary).

### Returned value

true - in case of successful check of the basic structures, otherwise - false.

### Note

Successful completion of the PositionModify(...) method does not always mean successful execution of the trade operation. It's necessary to check the result of trade request (trade server return code) using [ResultRetcode\(\)](#).

## PositionClose

Closes the position by specified symbol.

```
bool PositionClose(  
    const string  symbol,           // symbol  
    ulong        deviation=ULONG_MAX // deviation  
)
```

### Parameters

*symbol*

[in] Name of trade instrument, by which it is intended to close position.

*deviation=ULONG\_MAX*

[in] Maximal deviation from the current price (in points).

### Returned value

true - in case of successful check of the basic structures, otherwise - false.

### Note

Successful completion of the PositionClose(...) method does not always mean successful execution of the trade operation. It's necessary to check the result of trade request (trade server return code) using [ResultRetcode\(\)](#).



## Buy

Opens a long position with specified parameters.

```
bool Buy(  
    double      volume,           // position volume  
    const string symbol=NULL,     // symbol  
    double      price=0.0,        // price  
    double      sl=0.0,           // stop loss price  
    double      tp=0.0,           // take profit price  
    const string comment=""      // comment  
)
```

### Parameters

*volume*

[in] Position volume.

*symbol=NULL*

[in] Position symbol. If the symbol isn't specified, the current symbol will be used.

*price=0.0*

[in] Price. If the price isn't specified, the current market Ask price will be used.

*sl=0.0*

[in] Stop Loss price.

*tp=0.0*

[in] Take Profit price.

*comment=""*

[in] Comment.

### Returned value

true - in case of successful check of the structures, otherwise - false.

### Note

Successful completion of the Buy(...) method does not always mean successful execution of the trade operation. It's necessary to check the result of trade request (trade server [return\\_code](#)) using [ResultRetcode\(\)](#) and value, returned by [ResultDeal\(\)](#).

## Sell

Opens a short position with specified parameters.

```
bool Sell(  
    double      volume,           // position volume  
    const string symbol=NULL,     // symbol  
    double      price=0.0,        // price  
    double      sl=0.0,           // stop loss price  
    double      tp=0.0,           // take profit price  
    const string comment=""       // comment  
)
```

### Parameters

*volume*

[in] Position volume.

*symbol=NULL*

[in] Position symbol. If the symbol isn't specified, the current symbol will be used.

*price=0.0*

[in] Price. If the price isn't specified, the current market Bid price will be used.

*sl=0.0*

[in] Stop Loss price.

*tp=0.0*

[in] Take Profit price.

*comment=""*

[in] Comment.

### Returned value

true - in case of successful check of the structures, otherwise - false.

### Note

Successful completion of the Sell(...) method does not always mean successful execution of the trade operation. It's necessary to check the result of trade request (trade server [return\\_code](#)) using [ResultRetcode\(\)](#) and value, returned by [ResultDeal\(\)](#).

## BuyLimit

Places the pending order of Buy Limit type (buy at the price, lower than current market price) with specified parameters.

```
bool BuyLimit(
    double          volume,           // order volume
    double          price,            // order price
    const string    symbol=NULL,      // symbol
    double          sl=0.0,           // stop loss price
    double          tp=0.0,           // take profit price
    ENUM_ORDER_TYPE_TIME type_time=ORDER_TIME_GTC, // order lifetime
    datetime        expiration=0,     // order expiration time
    const string    comment=""        // comment
)
```

### Parameters

*volume*

[in] Order volume.

*price*

[in] Order price.

*symbol=NULL*

[in] Order symbol. If the symbol isn't specified, the current symbol will be used.

*sl=0.0*

[in] Stop Loss price.

*tp=0.0*

[in] Take Profit price.

*type\_time=ORDER\_TIME\_GTC*

[in] Order lifetime (value of [ENUM\\_ORDER\\_TYPE\\_TIME](#) enumeration).

*expiration=0*

[in] Order expiration time (used only if type\_time=[ORDER\\_TIME\\_SPECIFIED](#)).

*comment=""*

[in] Order comment.

### Returned value

true - in case of successful check of the structures, otherwise - false.

### Note

Successful completion of the BuyLimit(...) method does not always mean successful execution of the trade operation. It's necessary to check the result of trade request (trade server [return\\_code](#)) using [ResultRetcode\(\)](#) and value, returned by [ResultDeal\(\)](#).

## BuyStop

Places the pending order of Buy Stop type (buy at the price, higher than current market price) with specified parameters.

```
bool BuyStop(
    double          volume,           // order volume
    double          price,           // order price
    const string    symbol=NULL,      // symbol
    double          sl=0.0,          // stop loss price
    double          tp=0.0,          // take profit price
    ENUM_ORDER_TYPE_TIME type_time=ORDER_TIME_GTC, // order lifetime
    datetime        expiration=0,     // order expiration time
    const string    comment=""       // comment
)
```

### Parameters

*volume*

[in] Order volume.

*price*

[in] Order price.

*symbol=NULL*

[in] Order symbol. If the symbol isn't specified, the current symbol will be used.

*sl=0.0*

[in] Stop Loss price.

*tp=0.0*

[in] Take Profit price.

*type\_time=ORDER\_TIME\_GTC*

[in] Order lifetime (value of [ENUM\\_ORDER\\_TYPE\\_TIME](#) enumeration).

*expiration=0*

[in] Order expiration time (used only if type\_time=ORDER\_TIME\_SPECIFIED).

*comment=""*

[in] Order comment.

### Returned value

true - in case of successful check of the structures, otherwise - false.

### Note

Successful completion of the BuyStop(...) method does not always mean successful execution of the trade operation. It's necessary to check the result of trade request (trade server [return\\_code](#)) using [ResultRetcode\(\)](#) and value, returned by [ResultDeal\(\)](#).

## SellLimit

Places the pending order of Sell Limit type (sell at the price, higher than current market price) with specified parameters.

```
bool SellLimit(
    double          volume,           // order volume
    double          price,           // order price
    const string    symbol=NULL,     // symbol
    double          sl=0.0,          // stop loss price
    double          tp=0.0,          // take profit price
    ENUM_ORDER_TYPE_TIME type_time=ORDER_TIME_GTC, // order lifetime
    datetime        expiration=0,    // order expiration time
    const string    comment=""       // comment
)
```

### Parameters

*volume*

[in] Order volume.

*price*

[in] Order price.

*symbol=NULL*

[in] Order symbol. If the symbol isn't specified, the current symbol will be used.

*sl=0.0*

[in] Stop Loss price.

*tp=0.0*

[in] Take Profit price.

*type\_time=ORDER\_TIME\_GTC*

[in] Order lifetime (value of [ENUM\\_ORDER\\_TYPE\\_TIME](#) enumeration).

*expiration=0*

[in] Order expiration time (used only if type\_time=ORDER\_TIME\_SPECIFIED).

*comment=""*

[in] Order comment.

### Returned value

true - in case of successful check of the structures, otherwise - false.

### Note

Successful completion of the SellLimit(...) method does not always mean successful execution of the trade operation. It's necessary to check the result of trade request (trade server [return\\_code](#)) using [ResultRetcode\(\)](#) and value, returned by [ResultDeal\(\)](#).

## SellStop

Places the pending order of Buy Stop type (sell at the price, lower than current market price) with specified parameters.

```
bool SellStop(
    double          volume,           // order volume
    double          price,           // order price
    const string    symbol=NULL,     // symbol
    double          sl=0.0,          // stop loss price
    double          tp=0.0,          // take profit price
    ENUM_ORDER_TYPE_TIME type_time=ORDER_TIME_GTC, // order lifetime
    datetime        expiration=0,    // order expiration time
    const string    comment=""       // comment
)
```

### Parameters

*volume*

[in] Order volume.

*price*

[in] Order price.

*symbol=NULL*

[in] Order symbol. If the symbol isn't specified, the current symbol will be used.

*sl=0.0*

[in] Stop Loss price.

*tp=0.0*

[in] Take Profit price.

*type\_time=ORDER\_TIME\_GTC*

[in] Order lifetime (value of [ENUM\\_ORDER\\_TYPE\\_TIME](#) enumeration).

*expiration=0*

[in] Order expiration time (used only if type\_time=ORDER\_TIME\_SPECIFIED).

*comment=""*

[in] Order comment.

### Returned value

true - in case of successful check of the structures, otherwise - false.

### Note

Successful completion of the SellStop(...) method does not always mean successful execution of the trade operation. It's necessary to check the result of trade request (trade server [return\\_code](#)) using [ResultRetcode\(\)](#) and value, returned by [ResultDeal\(\)](#).

## Request

Gets the copy of the last request structure.

```
void Request(  
    MqlTradeRequest& request    // target structure  
    ) const
```

### Parameters

*request*

[out] Reference to the structure of [MqlTradeRequest](#) type.

### Returned value

None.

## RequestAction

Gets the trade operation type.

```
ENUM_TRADE_REQUEST_ACTIONS RequestAction() const
```

### Returned value

Trade operation type, used in the last request.



## RequestActionDescription

Gets the trade operation type as string.

```
string RequestActionDescription() const
```

### Returned value

Trade operation type (as string), used in the last request.

## RequestMagic

Gets the magic number of the Expert Advisor.

```
ulong RequestMagic() const
```

### Returned value

The magic number (ID) of the Expert Advisor, used in the last request.

## RequestOrder

Gets the order ticket, used in the last request.

```
ulong RequestOrder() const
```

### Returned value

Order ticket of the last request.

## RequestSymbol

Gets the name of the symbol, used in the last request.

```
string RequestSymbol() const
```

### Returned value

The name of the symbol, used in the last request.

## RequestVolume

Gets the trade volume (in lots), used in the last request.

```
double RequestVolume() const
```

### Returned value

The trade volume (in lots), used in the last request.

## RequestPrice

Gets the price, used in the last request.

```
double RequestPrice() const
```

### Returned value

Order price, used in the last request.

## RequestStopLimit

Gets the price of pending order of Stop Limit type, used in the last request.

```
double RequestStoplimit() const
```

### Returned value

The price of pending order of Stop Limit type, used in the last request.

## RequestSL

Gets the Stop Loss price of the order, used in the last request.

```
double RequestSL() const
```

### Returned value

The Stop Loss price, used in the last request.



## RequestTP

Gets the Take Profit price of the order, used in the last request.

```
double RequestTP() const
```

### Returned value

The Take Profit price, used in the last request.

## RequestDeviation

Gets the price deviation of the order, used in the last request.

```
ulong RequestDeviation() const
```

### Returned value

The price deviation of the order, used in the last request.

## RequestType

Gets the type of the order, used in the last request.

```
ENUM_ORDER_TYPE RequestType() const
```

### Returned value

Order type, used in the last request (value of [ENUM\\_ORDER\\_TYPE](#) enumeration).

## RequestTypeDescription

Gets the type of the order (as string) , used in the last request.

```
string RequestTypeDescription() const
```

### Returned value

The order type (as string), used in the last request.

## RequestTypeFilling

Gets the filling type of the order, used in the last request.

```
ENUM_ORDER_TYPE_FILLING RequestTypeFilling() const
```

### Returned value

The filling type of the order (value of [ENUM\\_ORDER\\_TYPE\\_FILLING](#)), used in the last request.

## RequestTypeFillingDescription

Gets the filling type of the order (as string), used in the last request.

```
string RequestTypeFillingDescription() const
```

### Returned value

The filling type (as string) of the order, used in the last request.

## RequestTypeTime

Gets the validity period of the order, used in the last request.

```
ENUM_ORDER_TYPE_TIME RequestTypeTime() const
```

### Returned value

The validity period of the order (value of [ENUM\\_ORDER\\_TYPE\\_TIME](#) enumeration), used in the last request.

## RequestTypeTimeDescription

Gets the validity period of the order (as string), used in the last request.

```
string RequestTypeTimeDescription() const
```

### Returned value

The validity period of the order (as string), used in the last request.



## RequestExpiration

Gets the expiration time of the order, used in the last request.

```
datetime RequestExpiration() const
```

### Returned value

The expiration time of the order, used in the last request.

## RequestComment

Gets the comment of the order, used in the last request.

```
string RequestComment() const
```

### Returned value

The comment of the order, used in the last request.

## Result

Gets the copy of the structure of the last request result.

```
void Result(  
    MqlTradeResult& result    // reference  
    ) const
```

### Parameters

*result*

[out] Reference to the structure of [MqlTradeRequest](#) type.

### Returned value

None.

## ResultRetcode

Gets the code of request result.

```
uint ResultRetcode() const
```

### Returned value

The [Code](#) of request result.

## ResultRetcodeDescription

Gets the code of request result as text.

```
string ResultRetcodeDescription() const
```

### Returned value

Code of the last request result as text.

## ResultDeal

Gets the deal ticket.

```
ulong ResultDeal() const
```

### Returned value

Deal ticket, if the deal is executed.

## ResultOrder

Gets the order ticket.

```
ulong ResultOrder() const
```

### Returned value

Order ticket, if the order is placed.

## ResultVolume

Gets the volume of deal or order.

```
double ResultVolume() const
```

### Returned value

Volume of deal or order.



## ResultPrice

Gets the price, confirmed by broker.

```
double ResultPrice() const
```

### Returned value

Price, confirmed by the broker.

## ResultBid

Gets the current bid price (the requote).

```
double ResultBid() const
```

### Returned value

Current bid price (the requote).

## ResultAsk

Gets the current ask price (the requote).

```
double ResultAsk() const
```

### Returned value

Current ask price (the requote).

## ResultComment

Gets the broker comment.

```
string ResultComment() const
```

### Returned value

Broker comment to the operation.

## CheckResult

Gets the copy of the structure of the last request check result.

```
void CheckResult(  
    MqlTradeCheckResult& check_result    // reference  
    ) const
```

### Parameters

*check\_result*

[out] Reference to the target structure of the [MqlTradeCheckResult](#) type.

### Returned value

None.

## CheckResultRetcode

Gets the value of the retcode field of [MqlTradeCheckResult](#) type, filled while checking of the request correctness.

```
uint CheckResultRetcode() const
```

### Returned value

The value of the retcode field (error code) of [MqlTradeCheckResult](#) type, filled while checking of the request correctness.

## CheckResultRetcodeDescription

Gets the string description of the retcode field of [MqlTradeCheckResult](#) type, filled while checking of the request correctness.

```
string ResultRetcodeDescription() const
```

### Returned value

The string description of the retcode field (Error code) of [MqlTradeCheckResult](#) type, filled while checking of the request correctness.

## CheckResultBalance

Gets the value of the balance field of [MqlTradeCheckResult](#) type, filled while checking of the request correctness.

```
double CheckResultBalance() const
```

### Returned value

The value of the balance field (balance value that will be after the execution of the trade operation) of [MqlTradeCheckResult](#) type, filled while checking of the request correctness.



## CheckResultEquity

Gets the value of the equity field of [MqlTradeCheckResult](#) type, filled while checking of the request correctness.

```
double CheckResultEquity() const
```

### Returned value

The value of the equity field (equity value that will be after the execution of the trade operation) of [MqlTradeCheckResult](#) type, filled while checking of the request correctness.

## CheckResultProfit

Gets the value of the profit field of [MqlTradeCheckResult](#) type, filled while checking of the request correctness.

```
double CheckResultProfit() const
```

### Returned value

The value of the profit field (profit value that will be after the execution of the trade operation) of [MqlTradeCheckResult](#) type, filled while checking of the request correctness.

## CheckResultMargin

Gets the value of the margin field of [MqlTradeCheckResult](#) type, filled while checking of the request correctness.

```
double CheckResultMargin() const
```

### Returned value

The value of the margin field (margin required for the trade operation) of [MqlTradeCheckResult](#) type, filled while checking of the request correctness.

## CheckResultMarginFree

Gets the value of the margin\_free field of [MqlTradeCheckResult](#) type, filled while checking of the request correctness.

```
double CheckResultMarginFree() const
```

### Returned value

The value of the margin\_free field (free margin that will be left after the execution of the trade operation) of [MqlTradeCheckResult](#) type, filled while checking of the request correctness.

## CheckResultMarginLevel

Gets the value of the margin\_level field of [MqlTradeCheckResult](#) type, filled while checking of the request correctness.

```
double CheckResultMarginLevel() const
```

### Returned value

The value of the margin\_level field (margin level that will be set after the execution of the trade operation) of [MqlTradeCheckResult](#) type, filled while checking of the request correctness.

## CheckResultComment

The value of the comment field of [MqlTradeCheckResult](#) type, filled while checking of the request correctness.

```
string CheckResultComment () const
```

### Returned value

The value of the comment field (Comment to the reply code, error description) of [MqlTradeCheckResult](#) type, filled while checking of the request correctness.

## PrintRequest

Prints the last request parameters into journal.

```
void PrintRequest() const
```

### Returned value

None.

## PrintResult

Prints the results of the last request into journal.

```
void PrintResult() const
```

### Returned value

None.



## FormatRequest

Prepares the formatted string with last request parameters.

```
string FormatRequest(  
    string&          str,          // target string  
    const MqlTradeRequest& request // request  
) const
```

### Parameters

*str*

[in] Target string, passed by reference.

*request*

[in] A structure of [MqlTradeRequest](#) type with parameters of the last request.

### Returned value

None.

## FormatRequestResult

Prepares the formatted string with results of the last request execution.

```
string FormatRequestResult(  
    string&          str,          // string  
    const MqlTradeRequest& request, // request structure  
    const MqlTradeResult& result   // result structure  
) const
```

### Parameters

*str*

[in] Target string, passed by reference.

*request*

[in] A structure of [MqlTradeRequest](#) type with parameters of the last request.

*result*

[in] A structure of [MqlTradeResult](#) type with results of the last request.

### Returned value

None.

## CTerminalInfo

CTerminalInfo is a class for simplified access to the properties of mql5 program environment.

### Description

CTerminalInfo class provides access to the properties of mql5 program environment.

### Declaration

```
class CTerminalInfo : public CObject
```

### Title

```
#include <Trade\TerminalInfo.mqh>
```

### Class methods by groups

| Methods for access to the properties of integer type |                                                                                                                         |
|------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------|
| <a href="#">Build</a>                                | Gets the build number of the client terminal                                                                            |
| <a href="#">IsConnected</a>                          | Gets the information about connection to trade server                                                                   |
| <a href="#">IsDLLsAllowed</a>                        | Gets the information about permission of DLL usage                                                                      |
| <a href="#">IsTradeAllowed</a>                       | Gets the information about permission to trade                                                                          |
| <a href="#">IsEmailEnabled</a>                       | Gets the information about permission to send e-mails to SMTP-server and login, specified in the terminal settings      |
| <a href="#">IsFtpEnabled</a>                         | Gets the information about permission to send trade reports to FTP server and login, specified in the terminal settings |
| <a href="#">MaxBars</a>                              | Gets the information about maximum number of bars on chart                                                              |
| <a href="#">CodePage</a>                             | Gets the information about the code page of the language in the client terminal                                         |
| <a href="#">CPUCores</a>                             | Gets the information about the CPU cores                                                                                |
| <a href="#">MemoryPhysical</a>                       | Gets the information about the physical memory (in Mb)                                                                  |
| <a href="#">MemoryTotal</a>                          | Gets the information about the total memory, available for the terminal/agent process (in Mb)                           |
| <a href="#">MemoryAvailable</a>                      | Gets the information about the free memory, available for the terminal/agent process (in Mb)                            |

|                                                            |                                                                                   |
|------------------------------------------------------------|-----------------------------------------------------------------------------------|
| <a href="#"><u>MemoryUsed</u></a>                          | Gets the information about the memory, used by the terminal/agent process (in Mb) |
| <a href="#"><u>IsX64</u></a>                               | Gets the information about the type of the client terminal (32/64 bit)            |
| <a href="#"><u>OpenCLSupport</u></a>                       | Gets the information about the version of OpenCL, supported by video card         |
| <a href="#"><u>DiskSpace</u></a>                           | Gets the information about free disk space (in Mb)                                |
| <b>Methods for access to the properties of string type</b> |                                                                                   |
| <a href="#"><u>Language</u></a>                            | Gets the language of the client terminal                                          |
| <a href="#"><u>Name</u></a>                                | Gets the name of the client terminal                                              |
| <a href="#"><u>Company</u></a>                             | Gets the company of the client terminal                                           |
| <a href="#"><u>Path</u></a>                                | Gets the folder of the client terminal                                            |
| <a href="#"><u>DataPath</u></a>                            | Gets the data folder of the client terminal                                       |
| <a href="#"><u>CommonDataPath</u></a>                      | Gets the common data folder of all client terminals, installed on the computer    |
| <b>Access to MQL5 API functions</b>                        |                                                                                   |
| <a href="#"><u>InfoInteger</u></a>                         | Gets the value of the property of integer type                                    |
| <a href="#"><u>InfoString</u></a>                          | Gets the value of property of string type                                         |

## Build

Gets the build number of the client terminal.

```
int CBuild() const
```

### Returned value

Build number of the client terminal.

### Note

To get the build number it uses the [TerminalInfoInteger\(\)](#) function ([TERMINAL\\_BUILD](#) property).

## IsConnected

Gets the information about connection to trade server.

```
bool IsConnected() const
```

### Returned value

true, if terminal is connected to trade server, otherwise false.

### Note

To get connection status it uses the [TerminalInfoInteger\(\)](#) function ([TERMINAL\\_CONNECTED](#) property).

## IsDLLsAllowed

Gets the information about permission of DLL usage.

```
bool IsDLLsAllowed() const
```

### Returned value

true, if DLL usage is allowed, otherwise false.

### Note

To get permission of DLL usage it uses the [TerminalInfoInteger\(\)](#) function ([TERMINAL\\_DLLS\\_ALLOWED](#) property).

## IsTradeAllowed

Gets the information about permission to trade.

```
bool IsTradeAllowed() const
```

### Returned value

true, if trade allowed, otherwise false.

### Note

To get permission to trade it uses the [TerminalInfoInteger\(\)](#) function ([TERMINAL\\_TRADE\\_ALLOWED](#) property).



## IsEmailEnabled

Gets the information about permission to send e-mails to SMTP-server and login, specified in the terminal settings.

```
bool IsEmailEnabled() const
```

### Returned value

true, if sending of e-mails is allowed, otherwise false.

### Note

To get the permission of e-mails sending it uses the [TerminalInfoInteger\(\)](#) function ([TERMINAL\\_EMAIL\\_ENABLED](#) property).

## IsFtpEnabled

Gets the information about permission to send trade reports to FTP server and login, specified in the terminal settings.

```
bool IsFtpEnabled() const
```

### Returned value

true, if trade reports sending to FTP server is allowed, otherwise false.

### Note

To get the information about permission to send trade reports it uses the [TerminalInfoInteger\(\)](#) function ([TERMINAL\\_FTP\\_ENABLED](#) property).

## MaxBars

Gets the maximum number of bars on chart, specified in the client terminal settings.

```
int MaxBars() const
```

### Returned value

Maximum number of bars on the chart.

### Note

To get the maximum number of bars on the chart it uses the [TerminalInfoInteger\(\)](#) function ([TERMINAL\\_MAXBARS](#) property).

## CodePage

Gets the information about code page of the language in the client terminal.

```
int CodePage () const
```

### Returned value

Code page of the language in the client terminal.

### Note

To get the code page it uses the [TerminalInfoInteger\(\)](#) function ([TERMINAL\\_CODEPAGE](#) property).

## CPUCores

Gets the information about the amount of CPU cores in the system.

```
int CPUCores () const
```

### Returned value

Amount of CPU cores in the system.

### Note

To get the amount of CPU cores it uses the [TerminalInfoInteger\(\)](#) function ([TERMINAL\\_CPU\\_CORES](#) property).

## MemoryPhysical

Gets the information about the physical memory (in Mb).

```
int MemoryPhysical() const
```

### Returned value

Physical memory (in Mb).

### Note

To get the physical memory it uses the [TerminalInfoInteger\(\)](#) function ([TERMINAL\\_MEMORY\\_PHYSICAL](#) property).

## MemoryTotal

Gets the information about the total memory, available for the client terminal/agent (in Mb).

```
int MemoryTotal() const
```

### Returned value

Total memory (in Mb), available for the terminal/agent.

### Note

To get the total memory it uses the [TerminalInfoInteger\(\)](#) function ([TERMINAL\\_MEMORY\\_TOTAL](#) property).

## MemoryAvailable

Gets the information about the free memory, available for the client terminal/agent (in Mb).

```
int MemoryTotal() const
```

### Returned value

Free memory (in Mb), available for the terminal/agent.

### Note

To get the free memory it uses the [TerminalInfoInteger\(\)](#) function ([TERMINAL\\_MEMORY\\_TOTAL](#) property).



## MemoryUsed

Gets the information about the memory, used by the client terminal/agent (in Mb).

```
int MemoryUsed() const
```

### Returned value

The memory, used by the client terminal/agent (in Mb).

### Note

To get the memory, used by the terminal it uses the [TerminalInfoInteger\(\)](#) function ([TERMINAL\\_MEMORY\\_USED](#) property).

## IsX64

Gets the information about the type of the client terminal.

```
bool IsX64() const
```

### Returned value

true, if 64-bit version is used, otherwise false.

### Note

To get the type of the client terminal it uses the [TerminalInfoInteger\(\)](#) function ([TERMINAL\\_X64](#) property).

## OpenCLSupport

Gets the information about the version of OpenCL, supported by video card.

```
int OpenCLSupport () const
```

### Returned value

The returned value has the following form: 0x00010002 = "1.2". The 0 means that OpenCL is not supported.

### Note

To get the version of OpenCL it uses the [TerminalInfoInteger\(\)](#) function ([TERMINAL\\_OPENCL\\_SUPPORT](#) property).

## DiskSpace

Gets the information about free disk space, available for the client terminal/agent (in Mb).

```
int MDiskSpace() const
```

### Returned value

Free disk space (in Mb), available for the client terminal/agent (for files, saved to MQL5\Files folder).

### Note

To get the free disk space it uses the [TerminalInfoInteger\(\)](#) function ([TERMINAL\\_DISK\\_SPACE](#) property).

## Language

Gets the information about the language in the client terminal.

```
string Language() const
```

### Returned value

Language, used in the client terminal.

### Note

To get the language it uses the [TerminalInfoString\(\)](#) function ([TERMINAL\\_LANGUAGE](#) property).

## Name

Gets the information of the name of the client terminal.

```
string Name() const
```

### Returned value

Name of the client terminal.

### Note

To get the name of the client terminal it uses the [TerminalInfoString\(\)](#) function ([TERMINAL\\_NAME](#) property).

## Company

Gets the information about the name of the broker.

```
string Company() const
```

### Returned value

The name of the broker.

### Note

To get the broker name it uses the [TerminalInfoString\(\)](#) function ([TERMINAL\\_COMPANY](#) property).

## Path

Gets the client terminal folder.

```
string Path() const
```

### Returned value

The client terminal folder.

### Note

To get the client terminal folder it uses the [TerminalInfoString\(\)](#) ([TERMINAL\\_PATH](#) property).



## DataPath

Gets the information about the terminal data folder.

```
string DataPath() const
```

### Returned value

Data folder of the client terminal.

### Note

To get the client terminal data folder it uses the [TerminalInfoString\(\)](#) function ([TERMINAL\\_DATA\\_PATH](#) property).

## CommonDataPath

Gets the common data folder of all client terminals, installed on the computer.

```
string CommonDataPath() const
```

### Returned value

Common data folder.

### Note

To get common data folder it uses the [TerminalInfoString\(\)](#) function ([COMMON\\_DATA\\_PATH](#) property).

## InfoInteger

Returns the value of a corresponding property of the mql5 program environment.

```
int TerminalInfoInteger(  
    int property_id    // identifier of a property  
);
```

### Parameters

*property\_id*

[in] Identifier of a property. Can be one of the values of the enumeration [ENUM\\_TERMINAL\\_INFO\\_INTEGER](#).

### Returned value

Value of int type.

### Note

To get the property value it uses the [TerminalInfoInteger\(\)](#) function.

## InfoString

the function returns the value of a corresponding property of the mql5 program environment. The property must be of string type.

```
string TerminalInfoString(  
    int property_id    // identifier of a property  
);
```

### Parameters

*property\_id*

[in] Identifier of a property. Perhaps one of the values of the enumeration [ENUM\\_TERMINAL\\_INFO\\_STRING](#).

### Returned value

Value of string type.

### Note

To get the property value it uses the [TerminalInfoString\(\)](#) function.

## Trading Strategy Classes

This section contains technical details of working with classes for creation and testing of trading strategies and description of the relevant components of the MQL5 standard library.

The use of these classes will save time when creating the trading strategies.

MQL5 Standard Library (in terms of trading strategies) is placed in the terminal directory, in the Include\Expert folder.

| Base classes                           | Description                                 |
|----------------------------------------|---------------------------------------------|
| <a href="#"><u>CExpertBase</u></a>     | Base class for all trading strategy classes |
| <a href="#"><u>CExpert</u></a>         | Base class for Expert Advisor               |
| <a href="#"><u>CExpertSignal</u></a>   | Base class for Trading Signal classes       |
| <a href="#"><u>CExpertTrailing</u></a> | Base class for Trailing Stop classes        |
| <a href="#"><u>CExpertMoney</u></a>    | Base class for Money Management classes     |

| Trading signal classes                   | Description                                                                                      |
|------------------------------------------|--------------------------------------------------------------------------------------------------|
| <a href="#"><u>CSignalAC</u></a>         | The module of signals based on market models of the indicator Accelerator Oscillator.            |
| <a href="#"><u>CSignalAMA</u></a>        | The module of signals based on market models of the indicator Adaptive Moving Average.           |
| <a href="#"><u>CSignalAO</u></a>         | The module of signals based on market models of the indicator Awesome Oscillator.                |
| <a href="#"><u>CSignalBearsPower</u></a> | The module of signals based on market models of the oscillator Bears Power.                      |
| <a href="#"><u>CSignalBullsPower</u></a> | The module of signals based on market models of the oscillator Bulls Power.                      |
| <a href="#"><u>CSignalCCI</u></a>        | The module of signals based on market models of the oscillator Commodity Channel Index.          |
| <a href="#"><u>CSignalDeM</u></a>        | The module of signals based on market models of the oscillator DeMarker.                         |
| <a href="#"><u>CSignalDEMA</u></a>       | The module of signals based on market models of the indicator Double Exponential Moving Average. |
| <a href="#"><u>CSignalEnvelopes</u></a>  | The module of signals based on market models of the indicator Envelopes.                         |
| <a href="#"><u>CSignalFrAMA</u></a>      | The module of signals based on market models of the indicator Fractal Adaptive Moving Average.   |
| <a href="#"><u>CSignalITF</u></a>        | The module of filtration of signals by time.                                                     |

| Trading signal classes              | Description                                                                                      |
|-------------------------------------|--------------------------------------------------------------------------------------------------|
| <a href="#"><u>CSignalMACD</u></a>  | The module of signals based on market models of the oscillator MACD.                             |
| <a href="#"><u>CSignalMA</u></a>    | The module of signals based on market models of the indicator Moving Average.                    |
| <a href="#"><u>CSignalSAR</u></a>   | The module of signals based on market models of the indicator Parabolic SAR.                     |
| <a href="#"><u>CSignalRSI</u></a>   | The module of signals based on market models of the oscillator Relative Strength Index.          |
| <a href="#"><u>CSignalRVI</u></a>   | The module of signals based on market models of the oscillator Relative Vigor Index.             |
| <a href="#"><u>CSignalStoch</u></a> | The module of signals based on market models of the oscillator Stochastic.                       |
| <a href="#"><u>CSignalTRIX</u></a>  | The module of signals based on market models of the oscillator Triple Exponential Average.       |
| <a href="#"><u>CSignalTEMA</u></a>  | The module of signals based on market models of the indicator Triple Exponential Moving Average. |
| <a href="#"><u>CSignalWPR</u></a>   | The module of signals based on market models of the oscillator Williams Percent Range.           |

| Trailing Stop classes                     | Description                                                                                   |
|-------------------------------------------|-----------------------------------------------------------------------------------------------|
| <a href="#"><u>CTrailingFixedPips</u></a> | This class implements Trailing Stop algorithm based on fixed points                           |
| <a href="#"><u>CTrailingMA</u></a>        | This class implements Trailing Stop algorithm based on the values of Moving Average indicator |
| <a href="#"><u>CTrailingNone</u></a>      | A gag class, it doesn't uses any Trailing Stop algorithm                                      |
| <a href="#"><u>CTrailingPSAR</u></a>      | This class implements Trailing Stop algorithm based on the values of Parabolic SAR indicator  |

| Money Management classes                 | Description                                                                 |
|------------------------------------------|-----------------------------------------------------------------------------|
| <a href="#"><u>CMoneyFixedLot</u></a>    | A class with an algorithm, based on trading with predefined fixed lot size. |
| <a href="#"><u>CMoneyFixedMargin</u></a> | A class with an algorithm, based on trading with predefined fixed margin.   |
| <a href="#"><u>CMoneyFixedRisk</u></a>   | A class with an algorithm, based on trading with predefined risk.           |

|                                            |                                                                                                                     |
|--------------------------------------------|---------------------------------------------------------------------------------------------------------------------|
| <a href="#"><u>CMoneyNone</u></a>          | A class with an algorithm, based on trading with minimal allowed lot size.                                          |
| <a href="#"><u>CMoneySizeOptimized</u></a> | A class with an algorithm, based on trading with variable lot size, depending on the results of the previous deals. |

## Base classes for Expert Advisors

This section contains technical details of working with classes for creation and testing of trading strategies and description of the relevant components of the MQL5 standard library.

The use of these classes will save time when creating the trading strategies.

MQL5 Standard Library (in terms of trading strategies) is placed in the terminal directory, in the Include\Expert folder.

| Class                                  | Description                                 |
|----------------------------------------|---------------------------------------------|
| <a href="#"><u>CExpertBase</u></a>     | Base class for all trading strategy classes |
| <a href="#"><u>CExpert</u></a>         | Base class for Expert Advisor               |
| <a href="#"><u>CExpertSignal</u></a>   | Base class for Trading Signal classes       |
| <a href="#"><u>CExpertTrailing</u></a> | Base class for Trailing Stop classes        |
| <a href="#"><u>CExpertMoney</u></a>    | Base class for Money Management classes     |



## CExpertBase

CExpertBase is a base class for the [CExpert](#) class and all trading strategy classes.

### Description

CExpertBase provides the data and methods, which are common to all objects of the Expert Advisor.

### Declaration

```
class CExpertBase : public CObject
```

### Title

```
#include <Expert\ExpertBase.mqh>
```

### Class Methods

#### Public Methods:

| Initialization                             |                                                         |
|--------------------------------------------|---------------------------------------------------------|
| virtual <a href="#">Init</a>               | Class instance initialization method                    |
| virtual <a href="#">ValidationSettings</a> | Checks the settings                                     |
| Parameters                                 |                                                         |
| <a href="#">Symbol</a>                     | Sets the symbol                                         |
| <a href="#">Period</a>                     | Sets the timeframe                                      |
| <a href="#">Magic</a>                      | Sets the Expert Advisor ID                              |
| Indicators and Timeseries                  |                                                         |
| virtual <a href="#">SetPriceSeries</a>     | Sets pointers to external timeseries (price series)     |
| virtual <a href="#">SetOtherSeries</a>     | Sets pointers to external timeseries (non-price series) |
| virtual <a href="#">InitIndicators</a>     | Initializes the indicators and timeseries               |
| Access to Protected Data                   |                                                         |
| <a href="#">InitPhase</a>                  | Gets the current phase of object initialization         |
| <a href="#">TrendType</a>                  | Sets trend type                                         |
| <a href="#">UsedSeries</a>                 | Gets bitmask of timeseries used                         |
| <a href="#">EveryTick</a>                  | Sets the "Every tick" flag                              |
| Access to Timeseries                       |                                                         |
| <a href="#">Open</a>                       | Gets the element of the Open timeseries by index        |

|                                   |                                                        |
|-----------------------------------|--------------------------------------------------------|
| <a href="#"><u>High</u></a>       | Gets the element of the High timeseries by index       |
| <a href="#"><u>Low</u></a>        | Gets the element of the Low timeseries by index        |
| <a href="#"><u>Close</u></a>      | Gets the element of the Close timeseries by index      |
| <a href="#"><u>Spread</u></a>     | Gets the element of the Spread timeseries by index     |
| <a href="#"><u>Time</u></a>       | Gets the element of the Time timeseries by index       |
| <a href="#"><u>TickVolume</u></a> | Gets the element of the TickVolume timeseries by index |
| <a href="#"><u>RealVolume</u></a> | Gets the element of the RealVolume timeseries by index |

### Protected Methods:

|                                               |                                                         |
|-----------------------------------------------|---------------------------------------------------------|
| <b>Initialization of Timeseries</b>           |                                                         |
| <a href="#"><u>InitOpen</u></a>               | Open timeseries initialization method                   |
| <a href="#"><u>InitHigh</u></a>               | High timeseries initialization method                   |
| <a href="#"><u>InitLow</u></a>                | Low timeseries initialization method                    |
| <a href="#"><u>InitClose</u></a>              | Close timeseries initialization method                  |
| <a href="#"><u>InitSpread</u></a>             | Spread timeseries initialization method                 |
| <a href="#"><u>InitTime</u></a>               | Time timeseries initialization method                   |
| <a href="#"><u>InitTickVolume</u></a>         | TickVolume timeseries initialization method             |
| <a href="#"><u>InitRealVolume</u></a>         | RealVolume timeseries initialization method             |
| <b>Service Methods</b>                        |                                                         |
| virtual <a href="#"><u>PriceLevelUnit</u></a> | Gets the price level unit                               |
| virtual <a href="#"><u>StartIndex</u></a>     | Gets the index of starting bar to analyze               |
| virtual <a href="#"><u>CompareMagic</u></a>   | Compares the Expert Advisor ID with the specified value |

## InitPhase

Gets the current phase of the object initialization.

```
ENUM_INIT_PHASE InitPhase()
```

### Returned value

Current phase of the object initialization.

### Note

The object initialization consist of several phases:

#### 1. Start initialization.

- start                      - after finish of the constructor
- finish                    - after successful completion of the [Init\(...\)](#) method.
- allowed                  - call of the [Init\(...\)](#) method
- not allowed              - call of the [ValidationSettings\(\)](#) method and other initialization methods

#### 2. Parameters setting phase. In this phase you need to set all the object parameters, used for creation of indicators.

- start                      - after successful completion of the [Init\(...\)](#) method
- finish                    - after successful completion of the [ValidationSettings\(\)](#) method
- allowed                  - call of [Symbol\(...\)](#) and [Period\(...\)](#) methods
- not allowed              - call of the [Init\(...\)](#), [SetPriceSeries\(...\)](#), [SetOtherSeries\(...\)](#) and [InitIndicators\(...\)](#) methods

#### 3. Checking of parameters.

- start                      - after successful completion of the [ValidationSettings\(\)](#) method
- finish                    - after successful completion of the [InitIndicators\(...\)](#) method
- allowed                  - call of the [Symbol\(...\)](#), [Period\(...\)](#) and [InitIndicators\(...\)](#) methods
- not allowed              - call of any other initialization methods

#### 4. Finish of initialization.

- start                      - after successful completion of the [InitIndicators\(...\)](#) method
- not allowed              - call of initialization methods

## TrendType

Sets trend type.

```
void TrendType(  
    M_TYPE_TREND value // new value  
)
```

### Parameters

*value*

[in] New value of trend type.

### Returned value

None.

## UsedSeries

Gets the bitmask of timeseries used.

```
int UsedSeries()
```

### Returned value

The list of used timeseries as bitmask.

### Note

If the bit is set, the corresponding timeseries is used, if it isn't set, the timeseries is not used.

The bit-timeseries correspondence:

- bit 0 - Open timeseries,
- bit 1 - High timeseries,
- bit 2 - Low timeseries,
- bit 3 - Close timeseries,
- bit 4 - Spread timeseries,
- bit 5 - Time timeseries,
- bit 6 - TickVolume timeseries,
- bit 7 - RealVolume timeseries.

## EveryTick

Sets the "Every tick" flag.

```
void EveryTick(  
    bool    value        // flag  
)
```

### Parameters

*value*

[in] New value of a flag.

### Returned value

None.

### Note

If the flag is not set, the processing method is called only at new bar on the working timeframe and symbol.

## Open

Gets the element of the Open timeseries by index.

```
double Open(  
    int ind // index  
)
```

### Parameters

*ind*

[in] Element index.

### Returned value

If successful, it returns the numerical value of the Open timeseries element with specified index, otherwise it returns EMPTY\_VALUE.

### Note

The EMPTY\_VALUE is returned in two cases:

1. Timeseries is not used (the corresponding bit is not set).
2. Element index is out of range.

## High

Gets the element of the High timeseries by index.

```
double High(  
    int ind // index  
)
```

### Parameters

*ind*

[in] Element index.

### Returned value

If successful, it returns the numerical value of the High timeseries element with specified index, otherwise it returns EMPTY\_VALUE.

### Note

The EMPTY\_VALUE is returned in two cases:

1. Timeseries is not used (the corresponding bit is not set).
2. Element index is out of range.



## Low

Gets the element of the Low timeseries by index.

```
double Low(  
    int ind // index  
)
```

### Parameters

*ind*

[in] Element index.

### Returned value

If successful, it returns the numerical value of the Low timeseries element with specified index, otherwise it returns EMPTY\_VALUE.

### Note

The EMPTY\_VALUE is returned in two cases:

1. Timeseries is not used (the corresponding bit is not set).
2. Element index is out of range.

## Close

Gets the element of the Close timeseries by index.

```
double Close (  
    int ind // index  
)
```

### Parameters

*ind*

[in] Element index.

### Returned value

If successful, it returns the numerical value of the Close timeseries element with specified index, otherwise it returns EMPTY\_VALUE.

### Note

The EMPTY\_VALUE is returned in two cases:

1. Timeseries is not used (the corresponding bit is not set).
2. Element index is out of range.

## Spread

Gets the element of the Spread timeseries by index.

```
double Spread(  
    int ind // index  
)
```

### Parameters

*ind*

[in] Element index.

### Returned value

If successful, it returns the numerical value of the Spread timeseries element with specified index, otherwise it returns EMPTY\_VALUE.

### Note

The EMPTY\_VALUE is returned in two cases:

1. Timeseries is not used (the corresponding bit is not set).
2. Element index is out of range.

## Time

Gets the element of the Time timeseries by index.

```
datetime Time(  
    int ind      // index  
)
```

### Parameters

*ind*

[in] Element index.

### Returned value

If successful, it returns the numerical value of the Time timeseries element with specified index, otherwise it returns EMPTY\_VALUE.

### Note

The EMPTY\_VALUE is returned in two cases:

1. Timeseries is not used (the corresponding bit is not set).
2. Element index is out of range.

## TickVolume

Gets the element of the TickVolume timeseries by index.

```
long TickVolume(  
    int ind // Index  
)
```

### Parameters

*ind*

[in] Element index.

### Returned value

If successful, it returns the numerical value of the TickVolume timeseries element with specified index, otherwise it returns EMPTY\_VALUE.

### Note

The EMPTY\_VALUE is returned in two cases:

1. Timeseries is not used (the corresponding bit is not set).
2. Element index is out of range.

## RealVolume

Gets the element of the RealVolume timeseries by index.

```
long RealVolume(  
    int ind // index  
)
```

### Parameters

*ind*

[in] Element index.

### Returned value

If successful, it returns the numerical value of the RealVolume timeseries element with specified index, otherwise it returns EMPTY\_VALUE.

### Note

The EMPTY\_VALUE is returned in two cases:

1. Timeseries is not used (the corresponding bit is not set).
2. Element index is out of range.

## Init

Initializes the object.

```
bool Init(  
    CSymbolInfo    symbol,    // symbol  
    ENUM_TIMEFRAMES period,    // timeframe  
    double         point      // point  
)
```

### Parameters

*symbol*

[in] Pointer to the object of [CSymbolInfo](#) type for access to symbol information.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

*point*

[in] The "weight" of 2/4-digit point.

### Returned value

true if successful, otherwise false.

## Symbol

Sets the symbol.

```
bool Symbol(  
    string name // symbol  
)
```

### Parameters

*name*

[in] Symbol.

### Returned value

true if successful, otherwise false.

### Note

The setting of working symbol is necessary if the object uses the symbol, different from symbol, defined at the initialization.



## Period

Sets the timeframe.

```
bool Period(  
    ENUM_TIMEFRAMES value    // timeframe  
)
```

### Parameters

*value*

[in] Timeframe.

### Returned value

true if successful, otherwise false.

### Note

The setting of working timeframe is necessary if the object uses the timeframe, different from timeframe, defined at the initialization.

## Magic

Sets the Expert Advisor ID.

```
void Magic(  
    ulong value    // magic  
)
```

### Parameters

*value*

[in] Expert Advisor ID.

### Returned value

None.

## ValidationSettings

Checks the settings.

```
virtual bool ValidationSettings()
```

### Returned value

true if successful, otherwise false.

## SetPriceSeries

Sets pointers to external price series.

```
virtual bool SetPriceSeries(  
    CiOpen*   open,      // pointer  
    CiHigh*   high,      // pointer  
    CiLow*    low,       // pointer  
    CiClose*  close      // pointer  
)
```

### Parameters

*open*

[in] Pointer to Open timeseries.

*high*

[in] Pointer to High timeseries.

*low*

[in] Pointer to Low timeseries.

*close*

[in] Pointer to Close timeseries.

### Returned value

true if successful, otherwise false.

### Note

The setting of pointers to external timeseries (price series) is necessary if the object uses timeseries of the symbol and timeframe, different from the symbol and timeframe, defined at initialization.

## SetOtherSeries

Sets pointers to external non-price series.

```
virtual bool SetOtherSeries(  
    CiSpread*      spread,      // pointer  
    CiTime*        time,        // pointer  
    CiTickVolume*  tick_volume, // pointer  
    CiRealVolume*  real_volume  // pointer  
)
```

### Parameters

*spread*

[in] Pointer to Spread timeseries.

*time*

[in] Pointer to Time timeseries.

*tick\_volume*

[in] Pointer to TickVolume timeseries.

*real\_volume*

[in] Pointer to RealVolume timeseries.

### Returned value

true if successful, otherwise false.

### Note

The setting of pointers to external timeseries (non-price series) is necessary if the object uses timeseries of the symbol and timeframe, different from the symbol and timeframe, defined at initialization.

## InitIndicators

Initializes all indicators and time series.

```
virtual bool InitIndicators(  
    CIndicators* indicators=NULL    // pointer  
)
```

### Parameters

*indicators*

[in] Pointer to collection of indicators and timeseries.

### Returned value

true if successful, otherwise false.

### Note

The timeseries are initialized only if the object uses the symbol or timeframe, different from the symbol or timeframe, defined at initialization.

## InitOpen

Initializes the Open timeseries.

```
bool InitOpen(  
    CIndicators* indicators // pointer  
)
```

### Parameters

*indicators*

[in] Pointer to collection of indicators and timeseries.

### Returned value

true if successful, otherwise false.

### Note

The Open timeseries is initialized only if Expert Advisor uses the symbol/timeframe, different from the symbol/timeframe defined at initialization (and timeseries is used further).

## InitHigh

Initializes the High timeseries.

```
bool InitHigh(  
    CIndicators* indicators // pointer  
)
```

### Parameters

*indicators*

[in] Pointer to collection of indicators and timeseries.

### Returned value

true if successful, otherwise false.

### Note

The High timeseries is initialized only if Expert Advisor uses the symbol/timeframe, different from the symbol/timeframe defined at initialization (and timeseries is used further).



## InitLow

Initializes the Low timeseries.

```
bool InitLow(  
    CIndicators* indicators // pointer  
)
```

### Parameters

*indicators*

[in] Pointer to collection of indicators and timeseries.

### Returned value

true if successful, otherwise false.

### Note

The Low timeseries is initialized only if Expert Advisor uses the symbol/timeframe, different from the symbol/timeframe defined at initialization (and timeseries is used further).

## InitClose

Initializes the Close timeseries.

```
bool InitClose(  
    CIndicators* indicators // pointer  
)
```

### Parameters

*indicators*

[in] Pointer to collection of indicators and timeseries.

### Returned value

true if successful, otherwise false.

### Note

The Close timeseries is initialized only if Expert Advisor uses the symbol/timeframe, different from the symbol/timeframe defined at initialization (and timeseries is used further).

## InitSpread

Initializes the Spread timeseries.

```
bool InitSpread(  
    CIndicators* indicators // pointer  
)
```

### Parameters

*indicators*

[in] Pointer to collection of indicators and timeseries.

### Returned value

true if successful, otherwise false.

### Note

The Spread timeseries is initialized only if Expert Advisor uses the symbol/timeframe, different from the symbol/timeframe defined at initialization (and timeseries is used further).

## InitTime

Initializes the Time timeseries.

```
bool InitTime(  
    CIndicators* indicators // pointer  
)
```

### Parameters

*indicators*

[in] Pointer to collection of indicators and timeseries.

### Returned value

true if successful, otherwise false.

### Note

The Time timeseries is initialized only if Expert Advisor uses the symbol/timeframe, different from the symbol/timeframe defined at initialization (and timeseries is used further).

## InitTickVolume

Initializes the TickVolume timeseries.

```
bool InitTickVolume(  
    CIndicators* indicators // pointer  
)
```

### Parameters

*indicators*

[in] Pointer to collection of indicators and timeseries.

### Returned value

true if successful, otherwise false.

### Note

The TickVolume timeseries is initialized only if Expert Advisor uses the symbol/timeframe, different from the symbol/timeframe defined at initialization (and timeseries is used further).

## InitRealVolume

Initializes the RealVolume timeseries.

```
bool InitRealVolume(  
    CIndicators* indicators // pointer  
)
```

### Parameters

*indicators*

[in] Pointer to collection of indicators and timeseries.

### Returned value

true if successful, otherwise false.

### Note

The RealVolume timeseries is initialized only if Expert Advisor uses the symbol/timeframe, different from the symbol/timeframe defined at initialization (and timeseries is used further).

## PriceLevelUnit

Gets the price level unit.

```
virtual double PriceLevelUnit ()
```

### Returned value

The value of Price Level unit.

### Note

The method of a base class returns the "weight" of the 2/4 digits point.

## StartIndex

Gets the index of starting bar to analyze.

```
virtual int StartIndex()
```

### Returned value

The index of starting bar to analyze.

### Note

The method returns 0 if the flag to analyze current bar is set to true (analysis from the current bar). If the flag is not set, it returns 1 (analysis from the last completed bar).



## CompareMagic

Compares the Expert Advisor ID (magic) with the specified value.

```
virtual bool CompareMagic(  
    ulong magic    // value to compare  
)
```

### Parameters

*magic*

[in] Value to compare.

### Returned value

true if they are equal, otherwise false.

## CExpert

CExpert is a base class for trading strategies. It has built-in algorithms for working with time series and indicators and a set of virtual methods for trading strategy.

How to use it:

1. Prepare an algorithm of the strategy;
2. Create your own class, inherited from CExpert class;
3. Override the virtual methods in your class with your own algorithms.

## Description

The CExpert class is a set of virtual methods for implementation of trading strategies.

## Declaration

```
class CExpert : public CExpertBase
```

## Title

```
#include <Expert\Expert.mqh>
```

## Class Methods

| Initialization                                    |                                         |
|---------------------------------------------------|-----------------------------------------|
| <a href="#"><u>Init</u></a>                       | Class instance initialization method    |
| virtual <a href="#"><u>InitSignal</u></a>         | Initializes Trading Signal object       |
| virtual <a href="#"><u>InitTrailing</u></a>       | Initializes Trailing Stop object        |
| virtual <a href="#"><u>InitMoney</u></a>          | Initializes Money Management object     |
| virtual <a href="#"><u>InitTrade</u></a>          | Initializes Trade object                |
| virtual <a href="#"><u>ValidationSettings</u></a> | Checks the settings                     |
| virtual <a href="#"><u>InitIndicators</u></a>     | Initializes indicators and timeseries   |
| virtual <a href="#"><u>InitParameters</u></a>     | Parameters initialization method        |
| virtual <a href="#"><u>Deinit</u></a>             | Class instance deinitialization method  |
| virtual <a href="#"><u>DeinitSignal</u></a>       | Deinitializes Trading Signal object     |
| virtual <a href="#"><u>DeinitTrailing</u></a>     | Deinitializes Trailing Stop object      |
| virtual <a href="#"><u>DeinitMoney</u></a>        | Deinitializes Money Management object   |
| virtual <a href="#"><u>DeinitTrade</u></a>        | Deinitializes Trade object              |
| virtual <a href="#"><u>DeinitIndicators</u></a>   | Deinitializes indicators and timeseries |
| <b>Parameters</b>                                 |                                         |
| <a href="#"><u>Magic</u></a>                      | Sets the Expert Advisor ID              |

|                                            |                                                   |
|--------------------------------------------|---------------------------------------------------|
| <a href="#"><u>MaxOrders</u></a>           | Gets/Sets the maximum amount of allowed orders    |
| <a href="#"><u>OnTickProcess</u></a>       | Sets a flag to proceed the "OnTick" event         |
| <a href="#"><u>OnTradeProcess</u></a>      | Sets a flag to proceed the "OnTrade" event        |
| <a href="#"><u>OnTimerProcess</u></a>      | Sets a flag to proceed the "OnTimer" event        |
| <a href="#"><u>OnChartEventProcess</u></a> | Sets a flag to proceed the "OnChartEvent" event   |
| <a href="#"><u>OnBookEventProcess</u></a>  | Sets a flag to proceed the "OnBookEvent" event    |
| <b>Event Processing Methods</b>            |                                                   |
| <a href="#"><u>OnTick</u></a>              | OnTick event handler                              |
| <a href="#"><u>OnTrade</u></a>             | OnTrade event handler                             |
| <a href="#"><u>OnTimer</u></a>             | OnTimer event handler                             |
| <a href="#"><u>OnChartEvent</u></a>        | OnChartEvent event handler                        |
| <a href="#"><u>OnBookEvent</u></a>         | OnBookEvent event handler                         |
| <b>Update Methods</b>                      |                                                   |
| <a href="#"><u>Refresh</u></a>             | Updates all data                                  |
| <b>Processing</b>                          |                                                   |
| <a href="#"><u>Processing</u></a>          | Main processing algorithm                         |
| <b>Market Entry Methods</b>                |                                                   |
| <a href="#"><u>CheckOpen</u></a>           | Checks position opening conditions                |
| <a href="#"><u>CheckOpenLong</u></a>       | Checks conditions to open long position           |
| <a href="#"><u>CheckOpenShort</u></a>      | Checks conditions to open short position          |
| <a href="#"><u>OpenLong</u></a>            | Opens a long position                             |
| <a href="#"><u>OpenShort</u></a>           | Opens a short position                            |
| <b>Market Exit Methods</b>                 |                                                   |
| <a href="#"><u>CheckClose</u></a>          | Checks conditions to close current position       |
| <a href="#"><u>CheckCloseLong</u></a>      | Checks conditions to close long position          |
| <a href="#"><u>CheckCloseShort</u></a>     | Checks conditions to close short position         |
| <a href="#"><u>CloseAll</u></a>            | Closes the opened position and deletes all orders |
| <a href="#"><u>Close</u></a>               | Closes the opened position                        |
| <a href="#"><u>CloseLong</u></a>           | Closes the long position                          |
| <a href="#"><u>CloseShort</u></a>          | Closes the short position                         |
| <b>Position Reverse Methods</b>            |                                                   |

|                                         |                                                                  |
|-----------------------------------------|------------------------------------------------------------------|
| <a href="#">CheckReverse</a>            | Checks conditions to reverse opened position                     |
| <a href="#">CheckReverseLong</a>        | Checks conditions to reverse long position                       |
| <a href="#">CheckReverseShort</a>       | Checks conditions to reverse short position                      |
| <a href="#">ReverseLong</a>             | Performs reverse operation of long position                      |
| <a href="#">ReverseShort</a>            | Performs reverse operation of short position                     |
| <b>Position/Order Trailing Methods</b>  |                                                                  |
| <a href="#">CheckTrailingStop</a>       | Checks conditions to modify position parameters                  |
| <a href="#">CheckTrailingStopLong</a>   | Checks Trailing Stop conditions of long position                 |
| <a href="#">CheckTrailingStopShort</a>  | Checks Trailing Stop conditions of short position                |
| <a href="#">TrailingStopLong</a>        | Performs Trailing Stop for long position                         |
| <a href="#">TrailingStopShort</a>       | Performs Trailing Stop for short position                        |
| <a href="#">CheckTrailingOrderLong</a>  | Checks Trailing Stop conditions of Buy Limit/Stop Pending order  |
| <a href="#">CheckTrailingOrderShort</a> | Checks Trailing Stop conditions of Sell Limit/Stop Pending order |
| <a href="#">TrailingOrderLong</a>       | Performs Trailing Stop for Buy Limit/Stop Pending order          |
| <a href="#">TrailingOrderShort</a>      | Performs Trailing Stop for Sell Limit/Stop Pending order         |
| <b>Order Delete Methods</b>             |                                                                  |
| <a href="#">CheckDeleteOrderLong</a>    | Checks conditions to delete Buy Pending order                    |
| <a href="#">CheckDeleteOrderShort</a>   | Checks conditions to delete Sell Pending order                   |
| <a href="#">DeleteOrders</a>            | Deletes all orders                                               |
| <a href="#">DeleteOrder</a>             | Deletes Stop/Limit Pending order                                 |
| <a href="#">DeleteOrderLong</a>         | Deletes Buy Limit/Stop pending order                             |
| <a href="#">DeleteOrderShort</a>        | Deletes Sell Limit/Stop pending order                            |
| <b>Trade Volume Methods</b>             |                                                                  |
| <a href="#">LotOpenLong</a>             | Gets trade volume for buy operation                              |
| <a href="#">LotOpenShort</a>            | Gets trade volume for sell operation                             |
| <a href="#">LotReverse</a>              | Gets trade volume for position reverse operation                 |
| <b>Trade History Methods</b>            |                                                                  |
| <a href="#">PrepareHistoryDate</a>      | Sets starting date for trade history tracking                    |

|                                                        |                                                                                                        |
|--------------------------------------------------------|--------------------------------------------------------------------------------------------------------|
| <a href="#"><u>HistoryPoint</u></a>                    | Creates a checkpoint of trade history (saves number of positions, orders, deals and historical orders) |
| <a href="#"><u>CheckTradeState</u></a>                 | Compares the current state with the saved one and calls the corresponding event handler                |
| <b>Event flags</b>                                     |                                                                                                        |
| <a href="#"><u>WaitEvent</u></a>                       | Sets the event waiting flag                                                                            |
| <a href="#"><u>NoWaitEvent</u></a>                     | Resets the event waiting flag                                                                          |
| <b>Trade Event Processing Methods</b>                  |                                                                                                        |
| <a href="#"><u>TradeEventPositionStopTake</u></a>      | Event handler of the "Position Stop Loss/Take Profit triggered" event                                  |
| <a href="#"><u>TradeEventOrderTriggered</u></a>        | Event handler of the "Pending Order Triggered" event                                                   |
| <a href="#"><u>TradeEventPositionOpened</u></a>        | Event handler of the "Position Opened" event                                                           |
| <a href="#"><u>TradeEventPositionVolumeChanged</u></a> | Event handler of the "Position Volume Changed" event                                                   |
| <a href="#"><u>TradeEventPositionModified</u></a>      | Event handler of the "Position Modified" event                                                         |
| <a href="#"><u>TradeEventPositionClosed</u></a>        | Event handler of the "Position Closed" event                                                           |
| <a href="#"><u>TradeEventOrderPlaced</u></a>           | Event handler of the "Pending Order Placed" event                                                      |
| <a href="#"><u>TradeEventOrderModified</u></a>         | Event handler of the "Pending Order Modified" event                                                    |
| <a href="#"><u>TradeEventOrderDeleted</u></a>          | Event handler of the "Pending Order Deleted" event                                                     |
| <a href="#"><u>TradeEventNotIdentified</u></a>         | Event handler of the non-identified event                                                              |
| <b>Service methods</b>                                 |                                                                                                        |
| <a href="#"><u>TimeframeAdd</u></a>                    | Adds a timeframe to track                                                                              |
| <a href="#"><u>TimeframesFlags</u></a>                 | Gets the flag indicating timeframes with a new bar                                                     |

## Init

Class instance initialization method.

```
bool Init(  
    string          symbol,          // symbol  
    ENUM_TIMEFRAMES period,          // timeframe  
    bool            every_tick,      // flag  
    ulong           magic            // magic  
)
```

### Parameters

*symbol*

[in] Symbol.

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

*every\_tick*

[in] Flag..

*magic*

[in] Expert Advisor ID (Magic number).

### Returned value

true if successful, otherwise false.

### Note

If *every\_tick* is set to true, the [Processing\(\)](#) method is called at each tick of the working symbol. otherwise, the [Processing\(\)](#) is called only when at new bar of the working symbol.

## Magic

Sets the Expert Advisor ID (magic).

```
void Magic(  
    ulong value    // new value  
)
```

### Parameters

*value*

[in] New value of Expert Advisor ID.

### Returned value

None.

### Note

It sets the value of Expert Advisor ID (magic) to the following classes: Trade, Signal, Money, Trailing.

### Implementation

```
//+-----+  
//| Sets magic number for object and its dependent objects |  
//| INPUT:  value - new value of magic number.           |  
//| OUTPUT: no.                                           |  
//| REMARK: no.                                           |  
//+-----+  
void CExpert::Magic(ulong value)  
{  
    if(m_trade!=NULL)    m_trade.SetExpertMagicNumber(value);  
    if(m_signal!=NULL)   m_signal.Magic(value);  
    if(m_money!=NULL)    m_money.Magic(value);  
    if(m_trailing!=NULL) m_trailing.Magic(value);  
//---  
    CExpertBase::Magic(value);  
}
```

## InitSignal

Initializes Trading Signal object.

```
virtual bool InitSignal(  
    CExpertSignal*    signal=NULL,        // pointer  
)
```

### Parameters

*signal*

[in] Pointer to [CExpertSignal](#) class object (or its heirs).

### Returned value

true if successful, otherwise false.

### Note

If signal is NULL, the [CExpertSignal](#) class will be used, it does nothing.



## InitTrailing

Initializes Trailing Stop object.

```
virtual bool InitTrailing(  
    CExpertTrailing* trailing=NULL,    // pointer  
)
```

### Parameters

*trailing*

[in] Pointer to [CExpertTrailing](#) class object (or its heir).

### Returned value

true if successful, otherwise false.

### Note

If trailing is NULL, the [ExpertTrailing](#) class will be used, it does nothing.

## InitMoney

Initializes the Money Management object.

```
virtual bool InitMoney(  
    CExpertMoney* money=NULL,      // pointer  
)
```

### Parameters

*money*

[in] Pointer to [CExpertMoney](#) class object (or its heir).

### Returned value

true if successful, otherwise false.

### Note

If money is NULL, the [CExpertMoney](#) class will be used, it uses the minimum lot.

## InitTrade

Initializes the Trade object.

```
virtual bool InitTrade(  
    ulong          magic,          // magic  
    CExpertTrade*  trade=NULL      // pointer  
)
```

### Parameters

*magic*

[in] Expert Advisor ID (will be used in trade requests).

*trade*

[in] Pointer to CExpertTrade object.

### Returned value

true if successful, otherwise false.

## Deinit

Class instance deinitialization method.

```
virtual void Deinit()
```

### Returned value

None.

## OnTickProcess

Sets a flag to proceed the [OnTick](#) event.

```
void OnTickOProcess(  
    bool    value    // flag  
)
```

### Parameters

*value*

[in] Flag to proceed the [OnTick](#) event.

### Returned value

None.

### Note

If the flag is true, the [OnTick](#) event is proceed, by default, the flag is set to true.

## OnTradeProcess

Sets a flag to proceed the [OnTrade](#) event.

```
void OnTradeProcess(  
    bool    value    // flag  
)
```

### Parameters

*value*

[in] Flag to proceed the [OnTrade](#) event.

### Returned value

None.

### Note

If the flag is true, the [OnTrade](#) event is proceed, by default, the flag is set to false.

## OnTimerProcess

Sets a flag to proceed the [OnTimer](#) event.

```
void OnTimerProcess(  
    bool    value    // flag  
)
```

### Parameters

*value*

[in] Flag to proceed the [OnTimer](#) event.

### Returned value

None.

### Note

If the flag is true, the [OnTimer](#) event is proceed, by default, the flag is set to false.

## OnChartEventProcess

Sets a flag to proceed the [OnChartEvent](#) event.

```
void OnChartEventProcess (
    bool    value    // flag
)
```

### Parameters

*value*

[in] Flag to proceed the [OnChartEvent](#) event.

### Returned value

None.

### Note

If the flag is true, the [OnChartEvent](#) event is proceed, by default, the flag is set to false.



## OnBookEventProcess

Sets a flag to proceed the [OnBookEvent](#) event.

```
void OnChartEventProcess (
    bool    value    // flag
)
```

### Parameters

*value*

[in] Flag to proceed the [OnBookEvent](#) event.

### Returned value

None.

### Note

If the flag is true, the [OnBookEvent](#) event is proceed, by default, the flag is set to false.

## MaxOrders (Get Method)

Gets the maximum amount of allowed orders.

```
int MaxOrders()
```

### Returned value

Maximum amount of allowed orders.

## MaxOrders (Set Method)

Sets the maximum amount of allowed orders.

```
void MaxOrders(  
    int max_orders // new value  
)
```

### Parameters

*max\_orders*

[in] New value of maximum amount of allowed orders.

### Returned value

None.

### Note

By default, the maximum amount of allowed orders = 1.

## Signal

Gets the pointer to the Trade Signal object.

```
CExpertSignal* Signal() const
```

### Returned value

Pointer to the Trade Signal object.

## ValidationSettings

Checks the settings.

```
virtual bool ValidationSettings()
```

### Returned value

true if successful, otherwise false.

### Note

It also checks the settings of all the Expert Advisor objects.

## InitIndicators

Initializes all indicators and timeseries.

```
virtual bool InitIndicators(  
    CIndicators* indicators=NULL    // pointer  
)
```

### Parameters

*indicators*

[in] Pointer to collection of indicators and timeseries.

### Returned value

true if successful, otherwise false.

### Note

The timeseries are initialized if the object uses the symbol or timeframe, different from the symbol or timeframe, defined at initialization. It calls consequentially InitIndicators() virtual methods of trading signal, trailing stop and money management objects.

## OnTick

Event handler of the [OnTick](#) event.

```
virtual void OnTick()
```

### Returned value

None.

## OnTrade

Event handler of the [OnTrade](#) event.

```
virtual void OnTrade()
```

### Returned value

None.

## OnTimer

Event handler of the [OnTimer](#) event.

```
virtual void OnTimer()
```

### Returned value

None.



## OnChartEvent

Event handler of the [OnChartEvent](#) event.

```
virtual void OnChartEvent(  
    const int      id,          // event id  
    const long&    lparam,      // long parameter  
    const double   dparam,      // double parameter  
    const string   sparam       // string parameter  
)
```

### Parameters

*id*

[in] Event ID.

*lparam*

[in] Event parameter of long type.

*dparam*

[in] Event parameter of double type.

*sparam*

[in] Event parameter of string type.

### Returned value

None.

## OnBookEvent

Event handler of the [OnBookEvent](#) event.

```
virtual void OnBookEvent(  
    const string&    symbol        // symbol  
)
```

### Parameters

*symbol*

[in] Symbol of [OnBookEvent](#) event.

### Returned value

None.

## InitParameters

Initializes parameters of Expert Advisor.

```
virtual bool InitParameters()
```

### Returned value

true if successful, otherwise false.

### Note

The InitParameters() function of [CExpert](#) base class does nothing and always returns true.

## DeinitTrade

Deinitializes Trade object.

```
virtual void DeinitTrade()
```

### Returned value

None.

## DeinitSignal

Deinitializes Signal object.

```
virtual void DeinitSignal ()
```

### Returned value

None.

## DeinitTrailing

Deinitializes Trailing object.

```
virtual void DeinitTrailing()
```

### Returned value

None.

## DeinitMoney

Deinitializes Money Management object.

```
virtual void DeinitMoney()
```

### Returned value

None.

## DeinitIndicators

Deinitializes all indicators and time series.

```
virtual void DeinitIndicators ()
```

### Returned value

None.

### Note

It also deinitializes all indicators and time series of all auxiliary objects.



## Refresh

Updates all data.

```
virtual bool Refresh()
```

### Returned value

true if further tick processing is needed, otherwise false.

### Note

It allows to determine the need of tick processing. If it needed, it updates all quotes and time series and indicators data and returns true.

### Implementation

```
//+-----+
//| Refreshing data for processing |
//| INPUT:  no.                    |
//| OUTPUT: true-if successful, false otherwise. |
//| REMARK: no.                    |
//+-----+
bool CExpert::Refresh()
{
    MqlDateTime time;
    //-- refresh rates
    if(!m_symbol.RefreshRates()) return(false);
    //-- check need processing
    TimeToStruct(m_symbol.Time(),time);
    if(m_period_flags!=WRONG_VALUE && m_period_flags!=0)
        if((m_period_flags & TimeframesFlags(time))==0) return(false);
    m_last_tick_time=time;
    //-- refresh indicators
    m_indicators.Refresh();
    //-- ok
    return(true);
}
```

## Processing

Main processing algorithm.

```
virtual bool Processing()
```

### Returned value

true if trade operation has been executed, otherwise false.

### Note

It does the following steps:

1. Checks the presence of the opened position on the symbol. If there isn't opened position, skip steps №2, №3 and №4.
2. Checks conditions to reverse opened position ([CheckReverse\(\)](#) method). If position has been "reversed", exit.
3. Checks conditions to close position ([CheckClose\(\)](#) method). If position has been closed, skip step №4.
4. Checks conditions to modify position parameters ([CheckTrailingStop\(\)](#) method). If position parameters has been modified, exit.
5. Check the presence of pending orders on the symbol. If there isn't any pending orders, go to step №9.
6. Checks condition to delete order ([CheckDeleteOrderLong\(\)](#) for buy pending orders or [CheckDeleteOrderShort\(\)](#) for sell pending orders). If the order has been deleted, go to step №9.
7. Check conditions to modify pending order parameters ([CheckTrailingOrderLong\(\)](#) for buy orders or [CheckTrailingOrderShort\(\)](#) for sell orders). If the order parameters has been modified, exit.
8. Exit.
9. Checks conditions to open position ([CheckOpen\(\)](#) method).

If you want to implement your own algorithm, you need to override the [Processing\(\)](#) method of the heir class.

### Implementation

```

//+-----+
//| Main function                                     |
//| INPUT:  no.                                       |
//| OUTPUT: true-if any trade operation processed, false otherwise. |
//| REMARK: no.                                       |
//+-----+
bool CExpert::Processing()
{
//--- check if open positions
    if(m_position.Select(m_symbol.Name()))
    {
        //--- open position is available
        //--- check the possibility of reverse the position
        if(CheckReverse()) return(true);
        //--- check the possibility of closing the position/delete pending orders
        if(!CheckClose())
        {
            //--- check the possibility of modifying the position
            if(CheckTrailingStop()) return(true);
            //--- return without operations
            return(false);
        }
    }
//--- check if placed pending orders
    int total=OrdersTotal();
    if(total!=0)
    {
        for(int i=total-1;i>=0;i--)
        {
            m_order.SelectByIndex(i);
            if(m_order.Symbol()!=m_symbol.Name()) continue;
            if(m_order.OrderType()==ORDER_TYPE_BUY_LIMIT || m_order.OrderType()==ORDER_T
            {
                //--- check the ability to delete a pending order to buy
                if(CheckDeleteOrderLong()) return(true);
                //--- check the possibility of modifying a pending order to buy
                if(CheckTrailingOrderLong()) return(true);
            }
            else
            {
                //--- check the ability to delete a pending order to sell
                if(CheckDeleteOrderShort()) return(true);
                //--- check the possibility of modifying a pending order to sell
                if(CheckTrailingOrderShort()) return(true);
            }
            //--- return without operations
            return(false);
        }
    }
//--- check the possibility of opening a position/setting pending order
    if(CheckOpen()) return(true);
//--- return without operations
    return(false);
}

```

## CheckOpen

Checks conditions to open position.

```
virtual bool CheckOpen()
```

### Returned value

true if any trade operation has been executed, otherwise false.

### Note

It checks conditions to open long ([CheckOpenLong\(\)](#)) and short ([CheckOpenShort\(\)](#)) positions.

### Implementation

```
//+-----+
//| Check for position open or limit/stop order set |
//| INPUT:  no. |
//| OUTPUT: true-if trade operation processed, false otherwise. |
//| REMARK: no. |
//+-----+
bool CExpert::CheckOpen()
{
    if(CheckOpenLong()) return(true);
    if(CheckOpenShort()) return(true);
    //--- return without operations
    return(false);
}
```

## CheckOpenLong

Checks conditions to open long position.

```
virtual bool CheckOpenLong()
```

### Returned value

true if any trade operation has been executed, otherwise false.

### Note

It checks conditions to open long position (CheckOpenLong() method of Signal object) and opens long position ([OpenLong\(\)](#) method) if necessary.

### Implementation

```
//+-----+
//| Check for long position open or limit/stop order set |
//| INPUT:  no. |
//| OUTPUT: true-if trade operation processed, false otherwise. |
//| REMARK: no. |
//+-----+
bool CExpert::CheckOpenLong()
{
    double price=EMPTY_VALUE;
    double sl=0.0;
    double tp=0.0;
    datetime expiration=TimeCurrent();
//--- check signal for long enter operations
    if(m_signal.CheckOpenLong(price,sl,tp,expiration))
    {
        if(!m_trade.SetOrderExpiration(expiration))
        {
            m_expiration=expiration;
        }
        return(OpenLong(price,sl,tp));
    }
//--- return without operations
    return(false);
}
```

## CheckOpenShort

Checks conditions to open short position.

```
virtual bool CheckOpenShort()
```

### Returned value

true if any trade operation has been executed, otherwise false.

### Note

It checks conditions to open short position (CheckOpenShort() method of Signal object) and opens a short position ([OpenShort\(\)](#) method) if necessary.

### Implementation

```
//+-----+
//| Check for short position open or limit/stop order set |
//| INPUT:  no. |
//| OUTPUT: true-if trade operation processed, false otherwise. |
//| REMARK: no. |
//+-----+
bool CExpert::CheckOpenShort()
{
    double price=EMPTY_VALUE;
    double sl=0.0;
    double tp=0.0;
    datetime expiration=TimeCurrent();
    //-- check signal for short enter operations
    if(m_signal.CheckOpenShort(price,sl,tp,expiration))
    {
        if(!m_trade.SetOrderExpiration(expiration))
        {
            m_expiration=expiration;
        }
        return(OpenShort(price,sl,tp));
    }
    //-- return without operations
    return(false);
}
```

## OpenLong

Opens a long position.

```
virtual bool  OpenLong(  
    double    price,      // price  
    double    sl,         // Stop Loss  
    double    tp          // Take Profit  
)
```

### Parameters

*price*

[in] Price.

*sl*

[in] Stop Loss price.

*tp*

[in] Take Profit price.

### Returned value

true if trade operation has been executed, otherwise false.

### Note

It gets trading volume ([LotOpenLong\(...\)](#) method) and opens a long position (Buy() method of Trade object) if trading volume is not equal to 0.

### Implementation

```
//+-----+  
//| Long position open or limit/stop order set |  
//| INPUT:  price - price,                    |  
//|         sl    - stop loss,                |  
//|         tp    - take profit.              |  
//| OUTPUT: true-if trade operation processed, false otherwise. |  
//| REMARK: no.                               |  
//+-----+  
bool CExpert::OpenLong(double price,double sl,double tp)  
{  
    if(price==EMPTY_VALUE) return(false);  
    //--- get lot for open  
    double lot=LotOpenLong(price,sl);  
    //--- check lot for open  
    if(lot==0.0) return(false);  
    //---  
    return(m_trade.Buy(lot,price,sl,tp));  
}
```

## OpenShort

Opens a short position.

```
virtual bool OpenShort(
    double price,    // price
    double sl,       // Stop Loss
    double tp        // Take Profit
)
```

### Parameters

*price*

[in] Price.

*sl*

[in] Stop Loss price.

*tp*

[in] Take Profit price.

### Returned value

true if trade operation has been executed, otherwise false.

### Note

It gets trading volume ([LotOpenShort\(\)](#) method) and opens a short position (by calling Sell method of Trade object) if trading volume is not equal to 0.

### Implementation

```
//+-----+
//| Short position open or limit/stop order set |
//| INPUT:  price - price,                      |
//|         sl    - stop loss,                  |
//|         tp    - take profit.                |
//| OUTPUT: true-if trade operation successful, false otherwise. |
//| REMARK: no.                                |
//+-----+
bool CExpert::OpenShort(double price,double sl,double tp)
{
    if(price==EMPTY_VALUE) return(false);
//--- get lot for open
    double lot=LotOpenShort(price,sl);
//--- check lot for open
    if(lot==0.0) return(false);
//---
    return(m_trade.Sell(lot,price,sl,tp));
}
```



## CheckReverse

Checks conditions to reverse opened position.

```
virtual bool CheckReverse()
```

### Returned value

true if any trade operation has been executed, otherwise false.

### Note

It checks conditions to reverse long ([CheckReverseLong\(\)](#)) and short ([CheckReverseShort\(\)](#)) positions.

### Implementation

```
//+-----+
//| Check for position reverse |
//| INPUT:  no.                |
//| OUTPUT: true-if trade operation processed, false otherwise. |
//| REMARK: no.                |
//+-----+
bool CExpert::CheckReverse()
{
    if(m_position.PositionType()==POSITION_TYPE_BUY)
    {
        //--- check the possibility of reverse the long position
        if(CheckReverseLong()) return(true);
    }
    else
        //--- check the possibility of reverse the short position
        if(CheckReverseShort()) return(true);
    //--- return without operations
    return(false);
}
```

## CheckReverseLong

Checks conditions to reverse long position.

```
virtual bool CheckReverseLong()
```

### Returned value

true if any trade operation has been executed, otherwise false.

### Note

It checks conditions to reverse long position (CheckReverseLong() method of Signal object) and perform reverse operation of the current long position ([ReverseLong\(...\)](#) method) if necessary.

### Implementation

```
//+-----+
//| Check for long position reverse |
//| INPUT:  no. |
//| OUTPUT: true-if trade operation processed, false otherwise. |
//| REMARK: no. |
//+-----+
bool CExpert::CheckReverseLong()
{
    double price=EMPTY_VALUE;
    double sl=0.0;
    double tp=0.0;
    datetime expiration=TimeCurrent();
    //-- check signal for long reverse operations
    if(m_signal.CheckReverseLong(price,sl,tp,expiration)) return(ReverseLong(price,sl,t
    //-- return without operations
    return(false);
}
```

## CheckReverseShort

Checks conditions to reverse short position.

```
virtual bool CheckReverseLong()
```

### Returned value

true if any trade operation has been executed, otherwise false.

### Note

It checks conditions to reverse short position (CheckReverseShort() method of Signal object) and perform reverse operation of the current short position ([ReverseShort\(\)](#) method) if necessary.

### Implementation

```
//+-----+
//| Check for short position reverse |
//| INPUT:  no.                      |
//| OUTPUT: true-if trade operation processed, false otherwise. |
//| REMARK: no.                      |
//+-----+
bool CExpert::CheckReverseShort()
{
    double price=EMPTY_VALUE;
    double sl=0.0;
    double tp=0.0;
    datetime expiration=TimeCurrent();
    //-- check signal for short reverse operations
    if(m_signal.CheckReverseShort(price,sl,tp,expiration)) return(ReverseShort(price,sl));
    //-- return without operations
    return(false);
}
```

## ReverseLong

Performs reverse operation of long position.

```
virtual bool ReverseLong(
    double price, // price
    double sl,    // Stop Loss
    double tp     // Take Profit
)
```

### Parameters

*price*

[in] Price.

*sl*

[in] Stop Loss price.

*tp*

[in] Take Profit price.

### Returned value

true if trade operation has been executed, otherwise false.

### Note

It gets the position reverse volume ([LotReverse\(\)](#) method) and perform trade operation of the long position reverse (Sell() method of Trade object) if trading volume is not equal to 0.

### Implementation

```
//+-----+
//| Long position reverse |
//| INPUT:  price - price, |
//|         sl    - stop loss, |
//|         tp    - take profit. |
//| OUTPUT: true-if trade operation processed, false otherwise. |
//| REMARK: no. |
//+-----+
bool CExpert::ReverseLong(double price,double sl,double tp)
{
    if(price==EMPTY_VALUE) return(false);
//--- get lot for reverse
    double lot=LotReverse(sl);
//--- check lot
    if(lot==0.0) return(false);
//---
    return(m_trade.Sell(lot,price,sl,tp));
}
```

## ReverseShort

Performs reverse operation of short position.

```
virtual bool ReverseShort(
    double price, // price
    double sl,    // Stop Loss
    double tp     // Take Profit
)
```

### Parameters

*price*

[in] Price.

*sl*

[in] Stop Loss price.

*tp*

[in] Take Profit price.

### Returned value

true if trade operation has been executed, otherwise false.

### Note

It gets position reverse volume ([LotReverse\(...\)](#) method) and perform trade operation of the short position reverse (Buy() method of Trade object) if trading volume is not equal to 0.

### Implementation

```
//+-----+
//| Short position reverse |
//| INPUT:  price - price,  |
//|         sl    - stop loss, |
//|         tp    - take profit. |
//| OUTPUT: true-if trade operation processed, false otherwise. |
//| REMARK: no. |
//+-----+
bool CExpert::ReverseShort(double price,double sl,double tp)
{
    if(price==EMPTY_VALUE) return(false);
//--- get lot for reverse
    double lot=LotReverse(sl);
//--- check lot
    if(lot==0.0) return(false);
//---
    return(m_trade.Buy(lot,price,sl,tp));
}
```

## CheckClose

Checks conditions to close position.

```
virtual bool CheckClose()
```

### Returned value

true if trade operation has been executed, otherwise false.

### Note

1. It checks Expert Advisor Stop Out conditions (CheckClose() method of money management object). If condition is satisfied, it closes the position and deletes all orders ([CloseAll\(...\)](#)) and exit.
2. It checks conditions to close long or short position ([CheckCloseLong\(\)](#) or [CheckCloseShort\(\)](#) methods) and if position is closed, it deletes all orders ([DeleteOrders\(\)](#) method).

### Implementation

```
//+-----+
//| Check for position close or limit/stop order delete |
//| INPUT:  no. |
//| OUTPUT: true-if trade operation processed, false otherwise. |
//| REMARK: no. |
//+-----+
bool CExpert::CheckClose()
{
    double lot;
    //-- position must be selected before call
    if ((lot=m_money.CheckClose(GetPointer(m_position)))!=0.0)
        return(CloseAll(lot));
    //-- check for position type
    if(m_position.PositionType()==POSITION_TYPE_BUY)
    {
        //-- check the possibility of closing the long position / delete pending orders
        if(CheckCloseLong())
        {
            DeleteOrders();
            return(true);
        }
    }
    else
    {
        //-- check the possibility of closing the short position / delete pending orders
        if(CheckCloseShort())
        {
            DeleteOrders();
            return(true);
        }
    }
    //-- return without operations
    return(false);
}
```

## CheckCloseLong

Checks conditions to close long position.

```
virtual bool CheckCloseLong()
```

### Returned value

true if trade operation has been executed, otherwise false.

### Note

It checks conditions to close long position (CheckCloseLong() method of Signal object) and if it satisfied, it closes the opened position ([CloseLong\(...\)](#) method).

### Implementation

```
//+-----+
//| Check for long position close or limit/stop order delete |
//| INPUT:  no. |
//| OUTPUT: true-if trade operation processed, false otherwise. |
//| REMARK: no. |
//+-----+
bool CExpert::CheckCloseLong()
{
    double price=EMPTY_VALUE;
    ///--- check for long close operations
    if(m_signal.CheckCloseLong(price))
        return(CloseLong(price));
    ///--- return without operations
    return(false);
}
```

## CheckCloseShort

Checks conditions to close short position.

```
virtual bool CheckCloseShort()
```

### Returned value

true if trade operation has been executed, otherwise false.

### Note

It checks conditions to close short position (CheckCloseShort() method of Signal object) and if it satisfied, it closes the position ( [CloseShort\(\)](#) method).

### Implementation

```
//+-----+
//| Check for short position close or limit/stop order delete |
//| INPUT:  no. |
//| OUTPUT: true-if trade operation processed, false otherwise. |
//| REMARK: no. |
//+-----+
bool CExpert::CheckCloseShort()
{
    double price=EMPTY_VALUE;
    ///--- check for short close operations
    if(m_signal.CheckCloseShort(price))
        return(CloseShort(price));
    ///--- return without operations
    return(false);
}
```



## CloseAll

It performs partial of full position closing.

```
virtual bool CloseAll(  
    double lot // lot  
)
```

### Parameters

*lot*

[in] Number of lots to reduce the position.

### Returned value

true if trade operation has been executed, otherwise false.

### Note

It performs partial of full position closing (Sell() and Buy() methods of CTrade class object for the long/short positions) and deletes all orders ([DeleteOrders\(\)](#) method).

### Implementation

```
//+-----+  
//| Position close and orders delete |  
//| INPUT: lot - volume for close. |  
//| OUTPUT: true-if trade operation processed, false otherwise. |  
//| REMARK: no. |  
//+-----+  
bool CExpert::CloseAll(double lot)  
{  
    bool result;  
    ///--- check for close operations  
    if(m_position.PositionType()==POSITION_TYPE_BUY) result=m_trade.Sell(lot,0,0,0);  
    else result=m_trade.Buy(lot,0,0,0);  
    result|=DeleteOrders();  
    ///---  
    return(result);  
}
```

## Close

Closes the opened position.

```
virtual bool Close()
```

### Returned value

true if trade operation has been executed, otherwise false.

### Note

Closes the position (PositionClose() method of CTrade class object).

### Implementation

```
//+-----+
//| Position close                                     |
//| INPUT:  no.                                         |
//| OUTPUT: true-if trade operation processed, false otherwise. |
//| REMARK: no.                                         |
//+-----+
bool CExpert::Close()
{
    return(m_trade.PositionClose(m_symbol.Name()));
}
```

## CloseLong

Closes the long position.

```
virtual bool CloseLong(  
    double price // price  
)
```

### Parameters

*price*

[in] Price.

### Returned value

true if trade operation has been executed, otherwise false.

### Note

Closes the long position (Sell(...) method of CTrade class object).

### Implementation

```
//+-----+  
//| Long position close |  
//| INPUT: price - price for close. |  
//| OUTPUT: true-if trade operation processed, false otherwise. |  
//| REMARK: no. |  
//+-----+  
bool CExpert::CloseLong(double price)  
{  
    if(price==EMPTY_VALUE) return(false);  
//---  
    return(m_trade.Sell(m_position.Volume(),price,0,0));  
}
```

## CloseShort

Closes the short position.

```
virtual bool CloseShort(  
    double price // price  
)
```

### Parameters

*price*

[in] Price.

### Returned value

true if trade operation has been executed, otherwise false.

### Note

Closes the short position (Buy(...) method of CTrade class object).

### Implementation

```
//+-----+  
//| Short position close |  
//| INPUT: price - price for close. |  
//| OUTPUT: true-if trade operation successful, false otherwise. |  
//| REMARK: no. |  
//+-----+  
bool CExpert::CloseShort(double price)  
{  
    if(price==EMPTY_VALUE) return(false);  
//---  
    return(m_trade.Buy(m_position.Volume(),price,0,0));  
}
```

## CheckTrailingStop

It checks Trailing Stop conditions of the opened position.

```
virtual bool CheckTrailingStop()
```

### Returned value

true if any trade operation has been executed, otherwise false.

### Note

It checks Trailing Stop conditions of the opened position ([CheckTrailingStopLong\(\)](#) or [CheckTrailingStopShort\(\)](#) for long and short positions).

### Implementation

```
//+-----+
//| Check for trailing stop/profit position          |
//| INPUT:  no.                                     |
//| OUTPUT: true-if trade operation processed, false otherwise. |
//| REMARK: no.                                     |
//+-----+
bool CExpert::CheckTrailingStop()
{
    //--- position must be selected before call
    if(m_position.PositionType()==POSITION_TYPE_BUY)
    {
        //--- check the possibility of modifying the long position
        if(CheckTrailingStopLong()) return(true);
    }
    else
    {
        //--- check the possibility of modifying the short position
        if(CheckTrailingStopShort()) return(true);
    }
    //--- return without operations
    return(false);
}
```

## CheckTrailingStopLong

It checks Trailing Stop conditions of the opened long position.

```
virtual bool CheckTrailingStopLong()
```

### Returned value

true if trade operation has been executed, otherwise false.

### Note

It checks Trailing Stop conditions of the opened long position (CheckTrailingStopLong(...) method of Expert Trailing object). If conditions are satisfied, it modifies the position parameters ([TrailingStopLong\(...\)](#) method).

### Implementation

```
//+-----+
//| Check for trailing stop/profit long position |
//| INPUT: no. |
//| OUTPUT: true-if trade operation processed, false otherwise. |
//| REMARK: no. |
//+-----+
bool CExpert::CheckTrailingStopLong()
{
    double sl=EMPTY_VALUE;
    double tp=EMPTY_VALUE;
    //-- check for long trailing stop operations
    if(m_trailing.CheckTrailingStopLong(GetPointer(m_position),sl,tp))
    {
        if(sl==EMPTY_VALUE) sl=m_position.StopLoss();
        if(tp==EMPTY_VALUE) tp=m_position.TakeProfit();
        //-- long trailing stop operations
        return(TrailingStopLong(sl,tp));
    }
    //-- return without operations
    return(false);
}
```

## CheckTrailingStopShort

It checks Trailing Stop conditions of the opened short position.

```
virtual bool CheckTrailingStopShort()
```

### Returned value

true if trade operation has been executed, otherwise false.

### Note

It checks Trailing Stop conditions of the opened short position (CheckTrailingStopShort(...) method of Expert Trailing object). If conditions are satisfied, it modifies the position parameters ([TrailingStopShort\(...\)](#) method).

### Implementation

```
//+-----+
//| Check for trailing stop/profit short position |
//| INPUT: no. |
//| OUTPUT: true-if trade operation processed, false otherwise. |
//| REMARK: no. |
//+-----+
bool CExpert::CheckTrailingStopShort()
{
    double sl=EMPTY_VALUE;
    double tp=EMPTY_VALUE;
    //-- check for short trailing stop operations
    if(m_trailing.CheckTrailingStopShort(GetPointer(m_position),sl,tp))
    {
        if(sl==EMPTY_VALUE) sl=m_position.StopLoss();
        if(tp==EMPTY_VALUE) tp=m_position.TakeProfit();
        //-- short trailing stop operations
        return(TrailingStopShort(sl,tp));
    }
    //-- return without operations
    return(false);
}
```

## TrailingStopLong

It modifies parameters of the opened long position.

```
virtual bool TrailingStopLong(  
    double sl,      // Stop Loss  
    double tp,      // Take Profit  
)
```

### Parameters

*sl*

[in] Stop Loss price.

*tp*

[in] Take Profit price.

### Returned value

true if trade operation has been executed, otherwise false.

### Note

The function modifies parameters of the opened long position (PositionModify(...) method of CTrade class object).

### Implementation

```
//+-----+  
//| Trailing stop/profit long position |  
//| INPUT:  sl - new stop loss,       |  
//|         tp - new take profit.     |  
//| OUTPUT: true-if trade operation successful, false otherwise. |  
//| REMARK: no.                       |  
//+-----+  
bool CExpert::TrailingStopLong(double sl,double tp)  
{  
    return(m_trade.PositionModify(m_symbol.Name(),sl,tp));  
}
```



## TrailingStopShort

It modifies parameters of the opened short position.

```
virtual bool TrailingStopLong(  
    double sl,    // Stop Loss  
    double tp,    // Take Profit  
)
```

### Parameters

*sl*

[in] Stop Loss price.

*tp*

[in] Take Profit price.

### Returned value

true if trade operation has been executed, otherwise false.

### Note

The function modifies parameters of the opened short position (PositionModify(...) method of CTrade class object).

### Implementation

```
//+-----+  
//| Trailing stop/profit short position |  
//| INPUT:  sl - new stop loss,         |  
//|         tp - new take profit.       |  
//| OUTPUT: true-if trade operation successful, false otherwise. |  
//| REMARK: no.                         |  
//+-----+  
bool CExpert::TrailingStopShort(double sl,double tp)  
{  
    return(m_trade.PositionModify(m_symbol.Name(),sl,tp));  
}
```

## CheckTrailingOrderLong

Checks Trailing Stop conditions of Buy Limit/Stop Pending order.

```
virtual bool CheckTrailingOrderLong()
```

### Returned value

true if trade operation has been executed, otherwise false.

### Note

It checks Trailing Stop conditions for buy limit/stop pending order (CheckTrailingOrderLong() method of Trade Signals object) and modifies the order parameters if necessary ([TrailingOrderLong\(...\)](#) method).

### Implementation

```
//+-----+
//| Check for trailing long limit/stop order |
//| INPUT:  no.                             |
//| OUTPUT: true-if trade operation processed, false otherwise. |
//| REMARK: no.                             |
//+-----+
bool CExpert::CheckTrailingOrderLong()
{
    double price;
    //--- check the possibility of modifying the long order
    if(m_signal.CheckTrailingOrderLong(GetPointer(m_order),price))
        return(TrailingOrderLong(m_order.PriceOpen()-price));
    //--- return without operations
    return(false);
}
```

## CheckTrailingOrderShort

It checks Trailing Stop conditions of Sell Limit/Stop pending order.

```
virtual bool CheckTrailingOrderShort()
```

### Returned value

true if trade operation has been executed, otherwise false.

### Note

It checks Trailing Stop conditions for sell limit/stop pending order (CheckTrailingOrderShort() method of Trade Signals object) and modifies the order parameters if necessary ([TrailingOrderShort\(\)](#) method).

### Implementation

```
//+-----+
//| Check for trailing short limit/stop order |
//| INPUT:  no.                               |
//| OUTPUT: true-if trade operation processed, false otherwise. |
//| REMARK: no.                               |
//+-----+
bool CExpert::CheckTrailingOrderShort()
{
    double price;
    //--- check the possibility of modifying the short order
    if(m_signal.CheckTrailingOrderShort(GetPointer(m_order),price))
        return(TrailingOrderShort(m_order.PriceOpen()-price));
    //--- return without operations
    return(false);
}
```

## TrailingOrderLong

It modifies parameters of Buy Limit/Stop Pending order.

```
virtual bool TrailingOrderLong(
    double delta // delta
)
```

### Parameters

*delta*

[in] Price delta.

### Returned value

true if trade operation has been executed, otherwise false.

### Note

It modifies parameters of Buy Limit/Stop Pending order (OrderModify(...) method of CTrade class object).

### Implementation

```
//+-----+
//| Trailing long limit/stop order |
//| INPUT: delta - price change. |
//| OUTPUT: true-if trade operation successful, false otherwise. |
//| REMARK: no. |
//+-----+
bool CExpert::TrailingOrderLong(double delta)
{
    ulong ticket=m_order.Ticket();
    double price =m_order.PriceOpen()-delta;
    double sl =m_order.StopLoss()-delta;
    double tp =m_order.TakeProfit()-delta;
    //--- modifying the long order
    return(m_trade.OrderModify(ticket,price,sl,tp,m_order.TypeTime(),m_order.TimeExpir
}
```

## TrailingOrderShort

It modifies parameters of Sell Limit/Stop Pending order.

```
virtual bool TrailingOrderShort(
    double delta // delta
)
```

### Parameters

*delta*

[in] Price delta.

### Returned value

true if trade operation has been executed, otherwise false.

### Note

It modifies parameters of Sell Limit/Stop Pending order (OrderModify(...) method of CTrade class object).

### Implementation

```
//+-----+
//| Trailing short limit/stop order |
//| INPUT: delta - price change. |
//| OUTPUT: true-if trade operation successful, false otherwise. |
//| REMARK: no. |
//+-----+
bool CExpert::TrailingOrderShort(double delta)
{
    ulong ticket=m_order.Ticket();
    double price =m_order.PriceOpen()-delta;
    double sl =m_order.StopLoss()-delta;
    double tp =m_order.TakeProfit()-delta;
    //--- modifying the short order
    return(m_trade.OrderModify(ticket,price,sl,tp,m_order.TypeTime(),m_order.TimeExpiration));
}
```

## CheckDeleteOrderLong

It checks conditions to delete Buy Limit/Stop Pending order.

```
virtual bool CheckDeleteOrderLong()
```

### Returned value

true if trade operation has been executed, otherwise false.

### Note

It checks the order expiration time. It checks conditions to delete the Sell Limit/Stop Pending order (CheckCloseLong(...) method of Signal class object) and deletes the order if condition is satisfied ( [DeleteOrderLong\(\)](#) method).

### Implementation

```
//+-----+
//| Check for delete long limit/stop order |
//| INPUT: no. |
//| OUTPUT: true-if trade operation processed, false otherwise. |
//| REMARK: no. |
//+-----+
bool CExpert::CheckDeleteOrderLong()
{
    double price;
    //-- check the possibility of deleting the long order
    if(m_expiration!=0 && TimeCurrent()>m_expiration)
    {
        m_expiration=0;
        return(DeleteOrderLong());
    }
    if(m_signal.CheckCloseLong(price))
        return(DeleteOrderLong());
    //-- return without operations
    return(false);
}
```

## CheckDeleteOrderShort

It checks conditions to delete Sell Limit/Stop Pending order.

```
virtual bool CheckDeleteOrderShort ()
```

### Returned value

true if trade operation has been executed, otherwise false.

### Note

It checks the order expiration time. It checks conditions to delete the Sell Limit/Stop Pending order (CheckCloseShort(...) method of Signal class object) and deletes the order if condition is satisfied ([DeleteOrderShort\(\)](#) method).

### Implementation

```
//+-----+
//| Check for delete short limit/stop order |
//| INPUT: no. |
//| OUTPUT: true-if trade operation processed, false otherwise. |
//| REMARK: no. |
//+-----+
bool CExpert::CheckDeleteOrderShort ()
{
    double price;
    //-- check the possibility of deleting the short order
    if(m_expiration!=0 && TimeCurrent()>m_expiration)
    {
        m_expiration=0;
        return(DeleteOrderShort());
    }
    if(m_signal.CheckCloseShort(price))
        return(DeleteOrderShort());
    //-- return without operations
    return(false);
}
```

## DeleteOrders

Deletes all orders.

```
virtual bool DeleteOrders()
```

### Returned value

true if trade operation has been executed, otherwise false.

### Note

It deletes all orders ([DeleteOrder\(\)](#) for all orders).

### Implementation

```
//+-----+
//| Delete all limit/stop orders |
//| INPUT: no. |
//| OUTPUT: true-if trade operation successful, false otherwise. |
//| REMARK: no. |
//+-----+
bool CExpert::DeleteOrders()
{
    bool result=false;
    int total=OrdersTotal();
    //---
    for(int i=total-1;i>=0;i--)
    {
        if(m_order.Select(OrderGetTicket(i)))
        {
            if(m_order.Symbol()!=m_symbol.Name()) continue;
            result|=DeleteOrder();
        }
    }
    //---
    return(result);
}
```



## DeleteOrder

Deletes the Limit/Stop Pending order.

```
virtual bool DeleteOrder()
```

### Returned value

true if trade operation has been executed, otherwise false.

### Note

It deletes the Limit/Stop Pending order (OrderDelete(...) method of CTrade class object).

### Implementation

```
//+-----+
//| Delete limit/stop order |
//| INPUT:  no.             |
//| OUTPUT: true-if trade operation successful, false otherwise. |
//| REMARK: no.             |
//+-----+
bool CExpert::DeleteOrder()
{
    return(m_trade.OrderDelete(m_order.Ticket()));
}
```

## DeleteOrderLong

Deletes the Buy Limit/Stop Pending order.

```
virtual bool DeleteOrderLong()
```

### Returned value

true if trade operation has been executed, otherwise false.

### Note

It deletes Buy Limit/Stop Pending order (OrderDelete(...) method of CTrade class object).

### Implementation

```
//+-----+
//| Delete long limit/stop order |
//| INPUT:  no.                  |
//| OUTPUT: true-if trade operation successful, false otherwise. |
//| REMARK: no.                  |
//+-----+
bool CExpert::DeleteOrderLong()
{
    return(m_trade.OrderDelete(m_order.Ticket()));
}
```

## DeleteOrderShort

Deletes the Sell Limit/Stop Pending order.

```
virtual bool DeleteOrderShort ()
```

### Returned value

true if trade operation has been executed, otherwise false.

### Note

It deletes the Sell Limit/Stop pending order (OrderDelete(...) method of CTrade class object).

### Implementation

```
//+-----+
//| Delete short limit/stop order |
//| INPUT:  no. |
//| OUTPUT: true-if trade operation successful, false otherwise. |
//| REMARK: no. |
//+-----+
bool CExpert::DeleteOrderShort ()
{
    return(m_trade.OrderDelete(m_order.Ticket()));
}
```

## LotOpenLong

Gets trade volume for buy operation.

```
double LotOpenLong(  
    double price, // price  
    double sl      // Stop Loss  
)
```

### Parameters

*price*

[in] Price.

*sl*

[in] Stop Loss price.

### Returned value

Trade volume (in lots) for buy operation.

### Note

It gets trade volume for buy operation (CheckOpenLong(...) method of money management object).

### Implementation

```
//+-----+  
//| Method of getting the lot for open long position. |  
//| INPUT:  price - price, |  
//|         sl    - stop loss. |  
//| OUTPUT: lot for open. |  
//| REMARK: no. |  
//+-----+  
double CExpert::LotOpenLong(double price,double sl)  
{  
    return(m_money.CheckOpenLong(price,sl));  
}
```

## LotOpenShort

Gets trade volume for sell operation.

```
double LotOpenShort(  
    double price, // price  
    double sl      // Stop Loss  
)
```

### Parameters

*price*

[in] Price.

*sl*

[in] Stop Loss price.

### Returned value

Trade volume (in lots) for sell operation.

### Note

It gets trade volume for sell operation (CheckOpenShort(...) method of money management object).

### Implementation

```
//+-----+  
//| Method of getting the lot for open short position. |  
//| INPUT:  price - price,                             |  
//|        sl    - stop loss.                         |  
//| OUTPUT: lot for open.                             |  
//| REMARK: no.                                       |  
//+-----+  
double CExpert::LotOpenShort(double price,double sl)  
{  
    return(m_money.CheckOpenShort(price,sl));  
}
```

## LotReverse

Gets trade volume for position reverse.

```
double LotReverse(  
    double    sl        // Stop Loss  
)
```

### Parameters

*sl*

[in] Stop Loss price.

### Returned value

Trade volume (in lots) for position reverse operation.

### Note

It gets trade volume for position reverse operation (CheckReverse(...) method of money management object).

### Implementation

```
//+-----+  
//| Method of getting the lot for reverse position. |  
//| INPUT:  sl - stop loss.                        |  
//| OUTPUT: lot for open.                          |  
//| REMARK: no.                                    |  
//+-----+  
double CExpert::LotReverse(double sl)  
{  
    return(m_money.CheckReverse(GetPointer(m_position),sl));  
}
```

## PrepareHistoryDate

Sets starting date for tracking of trade history.

```
void PrepareHistoryDate()
```

### Note

The trade history tracking period is set from the beginning of the month (but not less than one day).

## HistoryPoint

Creates a checkpoint of trade history (saves number of positions, orders, deals and historical orders).

```
void HistoryPoint(  
    bool    from_check_trade=false    // flag  
)
```

### Parameters

*from\_check\_trade=false*  
[in] Flag to avoid recursion.

### Note

It saves the amount of positions, orders, deals and historical orders.



## CheckTradeState

Compares the current state with the saved one and calls the corresponding event handler.

```
bool CheckTradeState()
```

### Returned value

true if event has been handled, otherwise - false.

### Note

It checks the number of positions, orders, deals and historical orders by comparing with the values, saved by [HistoryPoint\(\)](#) method. If trade history has changed, it calls the corresponding virtual event handler.

## WaitEvent

Sets the event waiting flag.

```
void WaitEvent(  
    ENUM_TRADE_EVENTS  event      // flag  
)
```

### Parameters

*event*

[in] Flag with events to set (ENUM\_TRADE\_EVENTS enumeration).

### Returned value

None.

### Event flags

```
///--- flags of expected events  
enum ENUM_TRADE_EVENTS  
{  
    TRADE_EVENT_NO_EVENT           =0,           // no expected events  
    TRADE_EVENT_POSITION_OPEN      =0x1,         // flag of expecting the "opening of  
    TRADE_EVENT_POSITION_VOLUME_CHANGE=0x2,      // flag of expecting of the "modifica  
    TRADE_EVENT_POSITION_MODIFY     =0x4,         // flag of expecting of the "modifica  
    TRADE_EVENT_POSITION_CLOSE      =0x8,         // flag of expecting of the "closing  
    TRADE_EVENT_POSITION_STOP_TAKE  =0x10,        // flag of expecting of the "triggeri  
    TRADE_EVENT_ORDER_PLACE         =0x20,        // flag of expecting of the "placing  
    TRADE_EVENT_ORDER_MODIFY        =0x40,        // flag of expecting of the "modifica  
    TRADE_EVENT_ORDER_DELETE        =0x80,        // flag of expecting of the "deletior  
    TRADE_EVENT_ORDER_TRIGGER       =0x100       // flag of expecting of the "triggeri  
};
```

## NoWaitEvent

Resets the event waiting flag.

```
void NoWaitEvent(  
    ENUM_TRADE_EVENTS    event    // flag  
)
```

### Parameters

*event*

[in] Flag with events to reset (ENUM\_TRADE\_EVENTS enumeration).

### Returned value

None.

### Event flags

```
///--- flags of expected events  
enum ENUM_TRADE_EVENTS  
{  
    TRADE_EVENT_NO_EVENT            =0,           // no expected events  
    TRADE_EVENT_POSITION_OPEN       =0x1,         // flag of expecting the "opening of  
    TRADE_EVENT_POSITION_VOLUME_CHANGE=0x2,       // flag of expecting of the "modifica  
    TRADE_EVENT_POSITION_MODIFY     =0x4,         // flag of expecting of the "modifica  
    TRADE_EVENT_POSITION_CLOSE      =0x8,         // flag of expecting of the "closing  
    TRADE_EVENT_POSITION_STOP_TAKE   =0x10,        // flag of expecting of the "triggeri  
    TRADE_EVENT_ORDER_PLACE         =0x20,        // flag of expecting of the "placing  
    TRADE_EVENT_ORDER_MODIFY        =0x40,        // flag of expecting of the "modifica  
    TRADE_EVENT_ORDER_DELETE        =0x80,        // flag of expecting of the "deletior  
    TRADE_EVENT_ORDER_TRIGGER       =0x100        // flag of expecting of the "triggeri  
};
```

## TradeEventPositionStopTake

Event handler of the "Position Stop Loss/Take Profit triggered" event.

```
virtual bool TradeEventPositionStopTake()
```

### Returned value

The [CExpert](#) class method does nothing and always returns true.

## TradeEventOrderTriggered

Event handler of the "Pending Order triggered" event.

```
virtual bool TradeEventOrderTriggered()
```

### Returned value

The [CExpert](#) class method does nothing and always returns true.

## TradeEventPositionOpened

Event handler of the "Position opened" event.

```
virtual bool TradeEventPositionOpened()
```

### Returned value

The [CExpert](#) class method does nothing and always returns true.

## TradeEventPositionVolumeChanged

Event handler of the "Position volume changed" event.

```
virtual bool TradeEventPositionVolumeChanged()
```

### Returned value

The [CExpert](#) class method does nothing and always returns true.

## TradeEventPositionModified

Event handler of the "Position modified" event.

```
virtual bool TradeEventPositionModified()
```

### Returned value

The [CExpert](#) class method does nothing and always returns true.



## TradeEventPositionClosed

Event handler of the "Position closed" event.

```
virtual bool TradeEventPositionClosed()
```

### Returned value

The [CExpert](#) class method does nothing and always returns true.

## TradeEventOrderPlaced

Event handler of the "Pending order placed" event.

```
virtual bool TradeEventOrderPlaced()
```

### Returned value

The [CExpert](#) class method does nothing and always returns true.

## TradeEventOrderModified

Event handler of the "Pending order modified" event.

```
virtual bool TradeEventOrderModified()
```

### Returned value

The [CEExpert](#) class method does nothing and always returns true.

## TradeEventOrderDeleted

Event handler of the "Pending order deleted" event.

```
virtual bool TradeEventOrderDeleted()
```

### Returned value

The [CExpert](#) class method does nothing and always returns true.

## TradeEventNotIdentified

Event handler of the non-identified event.

```
virtual bool TradeEventNotIdentified()
```

### Returned value

The [CEExpert](#) class method does nothing and always returns true.

### Note

Note that several trade events can arrive, in such cases it's difficult to identify them.

## TimeframeAdd

Add a timeframe for tracking.

```
void TimeframeAdd(  
    ENUM_TIMEFRAMES    period           // timeframe  
)
```

### Parameters

*period*

[in] Timeframe ([ENUM\\_TIMEFRAMES](#) enumeration).

### Returned value

None.

## TimeframesFlags

The method returns the flag indicating the timeframes with a new bar.

```
int TimeframesFlags(  
    MqlDateTime& time // variable for time  
)
```

### Parameters

*time*

[in] Variable of [MqlDateTime](#) type for new time, passed by reference.

### Returned value

It returns the flag, that indicates timeframes with a new bar.

## CExpertSignal

CExpertSignal is a base class for trading signals, it does nothing (except [CheckReverseLong\(\)](#) and [CheckReverseShort\(\)](#) methods) but provides the interfaces.

How to use it:

1. Prepare an algorithm for trading signals;
2. Create your own trading signal class, inherited from CExpertSignal class;
3. Override the virtual methods in your class with your own algorithms.

You can find an examples of trading signal classes in the Expert\Signal\ folder.

### Description

CExpertSignal is a base class for implementation of trading signal algorithms.

### Declaration

```
class CExpertSignal : public CExpertBase
```

### Title

```
#include <Expert\ExpertSignal.mqh>
```

### Class Methods

| Initialization                             |                                              |
|--------------------------------------------|----------------------------------------------|
| virtual <a href="#">InitIndicators</a>     | Initializes indicators and time series       |
| virtual <a href="#">ValidationSettings</a> | Checks the settings                          |
| virtual <a href="#">AddFilter</a>          | Adds a filter to combined signal             |
| Access to Protected Data                   |                                              |
| <a href="#">BasePrice</a>                  | Sets base price level                        |
| <a href="#">UsedSeries</a>                 | Gets the flags of timeseries used            |
| Parameters Setting                         |                                              |
| <a href="#">Weight</a>                     | Sets the value of "Weight" parameter         |
| <a href="#">PatternsUsage</a>              | Sets the value of "PatternsUsage" parameter  |
| <a href="#">General</a>                    | Sets the value of "General" parameter        |
| <a href="#">Ignore</a>                     | Sets the value of "Ignore" parameter         |
| <a href="#">Invert</a>                     | Sets the value of "Invert" parameter         |
| <a href="#">ThresholdOpen</a>              | Sets the value of "ThresholdOpen" parameter  |
| <a href="#">ThresholdClose</a>             | Sets the value of "ThresholdClose" parameter |
| <a href="#">PriceLevel</a>                 | Sets the value of "PriceLevel" parameter     |



|                                                        |                                                              |
|--------------------------------------------------------|--------------------------------------------------------------|
| <a href="#"><u>StopLevel</u></a>                       | Sets the value of "StopLevel" parameter                      |
| <a href="#"><u>TakeLevel</u></a>                       | Sets the value of "TakeLevel" parameter                      |
| <a href="#"><u>Expiration</u></a>                      | Sets the value of "Expiration" parameter                     |
| <a href="#"><u>Magic</u></a>                           | Sets the value of "Magic" parameter                          |
| <b>Checking Trading Conditions</b>                     |                                                              |
| virtual <a href="#"><u>CheckOpenLong</u></a>           | Checks conditions to open long position                      |
| virtual <a href="#"><u>CheckCloseLong</u></a>          | Checks conditions to close long position                     |
| virtual <a href="#"><u>CheckOpenShort</u></a>          | Checks conditions to open short position                     |
| virtual <a href="#"><u>CheckCloseShort</u></a>         | Checks conditions to close short position                    |
| virtual <a href="#"><u>CheckReverseLong</u></a>        | Checks conditions of long position reversal                  |
| virtual <a href="#"><u>CheckReverseShort</u></a>       | Checks conditions of short position reversal                 |
| <b>Trade Parameters Setting</b>                        |                                                              |
| virtual <a href="#"><u>OpenLongParams</u></a>          | Sets parameters for long position opening                    |
| virtual <a href="#"><u>OpenShortParams</u></a>         | Sets parameters for short position opening                   |
| virtual <a href="#"><u>CloseLongParams</u></a>         | Sets parameters for long position closing                    |
| virtual <a href="#"><u>CloseShortParams</u></a>        | Sets parameters for short position closing                   |
| <b>Checking of Order Trailing Conditions</b>           |                                                              |
| virtual <a href="#"><u>CheckTrailingOrderLong</u></a>  | Checks conditions to modify parameters of Buy Pending order  |
| virtual <a href="#"><u>CheckTrailingOrderShort</u></a> | Checks conditions to modify parameters of Sell Pending order |
| <b>Methods to Check Formation of Market Orders</b>     |                                                              |
| virtual <a href="#"><u>LongCondition</u></a>           | Gets the result of checking of buy conditions                |
| virtual <a href="#"><u>ShortCondition</u></a>          | Gets the result of checking of sell conditions               |
| virtual <a href="#"><u>Direction</u></a>               | Gets the "weighted" direction of price                       |

## BasePrice

Sets base price level.

```
void BasePrice(  
    double    value        // new value  
)
```

### Parameters

*value*

[in] New value of Base price level.

### Returned value

None.

## UsedSeries

Gets the flags of the timeseries used.

```
int BasePrice()
```

### Returned value

Flags of the used timeseries (if the symbol/timeframe corresponds to the working symbol/timeframe), otherwise 0.

## Weight

Sets new value of "Weight" parameter.

```
void Weight(  
    double    value        // new value  
)
```

### Parameters

*value*

[in] New value of "Weight".

### Returned value

None.

## PatternUsage

Sets new value of "PatternsUsage" parameter.

```
void PatternUsage(  
    double    value        // new value  
)
```

### Parameters

*value*

[in] New value of "PatternsUsage".

### Returned value

None.

## General

Sets new value of "General" parameter.

```
void General (
    int    value      // new value
)
```

### Parameters

*value*

[in] New value of "General".

### Returned value

None.

## Ignore

Sets new value of "Ignore" parameter.

```
void Ignore(  
    long    value        // new value  
)
```

### Parameters

*value*

[in] New value of "Ignore".

### Returned value

None.

## Invert

Sets new value of "Invert" parameter.

```
void Invert(  
    long    value    // new value  
)
```

### Parameters

*value*

[in] New value of "Invert".

### Returned value

None.



## ThresholdOpen

Sets new value of "ThresholdOpen" parameter.

```
void ThresholdOpen(  
    long    value    // new value  
)
```

### Parameters

*value*

[in] New value of "ThresholdOpen".

### Returned value

None.

### Note

The range of "ThresholdOpen" parameter is from 0 to 100. Used when "voting" to open position.

## ThresholdClose

Sets new value of "ThresholdClose" parameter.

```
void ThresholdOpen(  
    long    value        // new value  
)
```

### Parameters

*value*

[in] New value of "ThresholdClose".

### Returned value

None.

### Note

The range of "ThresholdClose" parameter is from 0 to 100. Used when "voting" to close position.

## PriceLevel

Sets new value of "PriceLevel" parameter.

```
void PriceLevel(  
    double    value        // new value  
)
```

### Parameters

*value*

[in] New value of "PriceLevel".

### Returned value

None.

### Note

The value of "PriceLevel" is defined in price level units. The numerical values of price level unit is returned by [PriceLevelUnit\(\)](#) method. The "PriceLevel" is used to define the open price relative to the base price.

## StopLevel

Sets new value of "StopLevel" parameter.

```
void StopLevel(  
    double    value        // new value  
)
```

### Parameters

*value*

[in] New value of "StopLevel".

### Returned value

None.

### Note

The value of "StopLevel" is defined in price level units. The numerical values of price level unit is returned by [PriceLevelUnit\(\)](#) method. The "StopLevel" is used to define the Stop Loss price relative to the open price.

## TakeLevel

Sets new value of "TakeLevel" parameter.

```
void TakeLevel(  
    double    value        // new value  
)
```

### Parameters

*value*

[in] New value of "TakeLevel".

### Returned value

None.

### Note

The value of "TakeLevel" is defined in price level units. The numerical values of price level unit is returned by [PriceLevelUnit\(\)](#) method. The "TakeLevel" is used to define the Take Profit price relative to the open price.

## Expiration

Sets the value of "Expiration" parameter.

```
void Expiration(  
    int    value        // new value  
)
```

### Parameters

*value*

[in] New value of "Expiration".

### Returned value

None.

### Note

The value of "Expiration" parameter is defined in bars. It used as Expiration time for Pending Orders (when trading using pending orders).

## Magic

Sets the value of "Magic" parameter.

```
void Magic(  
    int    value        // new value  
)
```

### Parameters

*value*

[in] New value of "Magic" (Expert Advisor ID).

### Returned value

None.

## ValidationSettings

Checks the settings.

```
virtual bool ValidationSettings()
```

### Returned value

true if successful, otherwise false.



## InitIndicators

Initializes all indicators and time series.

```
virtual bool InitIndicators(  
    CIndicators* indicators // pointer  
)
```

### Parameters

*indicators*

[in] Pointer to collection of indicators and timeseries.

### Returned value

true if successful, otherwise false.

### Note

The timeseries are initialized only if the object uses the symbol or timeframe, different from the symbol or timeframe, defined at initialization.

## AddFilter

Adds a filter to the composite signal.

```
virtual bool InitIndicators(  
    CExpertSignal* filter    // pointer  
)
```

### Parameters

*indicators*

[in] Pointer to filter object.

### Returned value

true if successful, otherwise false.

## CheckOpenLong

Checks conditions to open long position.

```
virtual bool CheckOpenLong(  
    double& price,           // price  
    double& sl,              // Stop Loss  
    double& tp,              // Take Profit  
    datetime& expiration    // expiration  
)
```

### Parameters

*price*

[in][out] Variable for price, passed by reference.

*sl*

[in][out] Variable for Stop Loss price, passed by reference.

*tp*

[in][out] Variable for Take Profit price, passed by reference.

*expiration*

[in][out] Variable for expiration time, passed by reference.

### Returned value

true if condition is satisfied, otherwise false.

## CheckOpenShort

Checks conditions to open short position.

```
virtual bool CheckOpenShort(  
    double& price,           // price  
    double& sl,              // Stop Loss  
    double& tp,              // Take Profit  
    datetime& expiration    // expiration  
)
```

### Parameters

*price*

[in][out] Variable for price, passed by reference.

*sl*

[in][out] Variable for Stop Loss price, passed by reference.

*tp*

[in][out] Variable for Take Profit price, passed by reference.

*expiration*

[in][out] Variable for expiration time, passed by reference.

### Returned value

true if condition is satisfied, otherwise false.

## OpenLongParams

Sets parameters to open long position.

```
virtual bool OpenLongParams(  
    double& price,           // price  
    double& sl,              // Stop Loss  
    double& tp,              // Take Profit  
    datetime& expiration    // expiration  
)
```

### Parameters

*price*

[in][out] Variable for price, passed by reference.

*sl*

[in][out] Variable for Stop Loss price, passed by reference.

*tp*

[in][out] Variable for Take Profit price, passed by reference.

*expiration*

[in][out] Variable for expiration time, passed by reference.

### Returned value

true if successful, otherwise false.

## OpenShortParams

Sets parameters to open short position.

```
virtual bool OpenShortParams(  
    double& price,           // price  
    double& sl,              // Stop Loss  
    double& tp,              // Take Profit  
    datetime& expiration     // expiration  
)
```

### Parameters

*price*

[in][out] Variable for price, passed by reference.

*sl*

[in][out] Variable for Stop Loss price, passed by reference.

*tp*

[in][out] Variable for Take Profit price, passed by reference.

*expiration*

[in][out] Variable for expiration time, passed by reference.

### Returned value

true if successful, otherwise false.

## CheckCloseLong

Checks conditions to close long position.

```
virtual bool CheckCloseLong(  
    double& price    // price  
)
```

### Parameters

*price*

[in][out] Variable for close price, passed by reference.

### Returned value

true if condition is satisfied, otherwise false.

## CheckCloseShort

Checks conditions to close short position.

```
virtual bool CheckCloseShort(  
    double& price    // price  
)
```

### Parameters

*price*

[in][out] Variable for close price, passed by reference.

### Returned value

true if condition is satisfied, otherwise false.



## CloseLongParams

Sets parameters to close long position.

```
virtual bool CloseLongParams(  
    double& price // price  
)
```

### Parameters

*price*

[in][out] Variable for close price, passed by reference.

### Returned value

true if successful, otherwise false.

## CloseShortParams

Sets parameters to close short position.

```
virtual bool CloseShortParams (  
    double& price // price  
)
```

### Parameters

*price*

[in][out] Variable for close price, passed by reference.

### Returned value

true if successful, otherwise false.

## CheckReverseLong

Checks conditions of long position reversal.

```
virtual bool CheckReverseLong(  
    double& price,           // price  
    double& sl,              // Stop Loss  
    double& tp,              // Take Profit  
    datetime& expiration    // expiration  
)
```

### Parameters

*price*

[in][out] Variable for price, passed by reference.

*sl*

[in][out] Variable for Stop Loss price, passed by reference.

*tp*

[in][out] Variable for Take Profit price, passed by reference.

*expiration*

[in][out] Variable for expiration time, passed by reference.

### Returned value

true if condition is satisfied, otherwise false.

## CheckReverseShort

Checks conditions of short position reversal.

```
virtual bool CheckReverseShort(  
    double& price,           // price  
    double& sl,              // Stop Loss  
    double& tp,              // Take Profit  
    datetime& expiration    // expiration  
)
```

### Parameters

*price*

[in][out] Variable for reversal price, passed by reference.

*sl*

[in][out] Variable for Stop Loss price, passed by reference.

*tp*

[in][out] Variable for Take Profit price, passed by reference.

*expiration*

[in][out] Variable for expiration time, passed by reference.

### Returned value

true if condition is satisfied, otherwise false.

## CheckTrailingOrderLong

Checks conditions to modify parameters of Buy Pending order.

```
virtual bool CheckTrailingOrderLong(  
    COrderInfo*    order,          // order  
    double&        price           // price  
)
```

### Parameters

*order*

[in] Pointer to [COrderInfo](#) class object.

*price*

[in][out] Variable for Stop Loss price.

### Returned value

true if condition is satisfied, otherwise false.

## CheckTrailingOrderShort

Checks conditions to modify parameters of Sell Pending order.

```
virtual bool CheckTrailingOrderShort(  
    COrderInfo*    order,          // order  
    double&        price           // price  
)
```

### Parameters

*order*

[in] Pointer to [COrderInfo](#) class object.

*price*

[in][out] Variable for Stop Loss price.

### Returned value

true if condition is satisfied, otherwise false.

## LongCondition

Checks conditions to open long position.

```
virtual int LongCondition()
```

### Returned value

Then conditions are satisfied, it returns the value from 1 to 100 (depending on "strength" of a signal), if there isn't a signal to open long position, it returns 0.

### Note

The LongCondition() method of a base class has no implementation of checking of conditions to open long position and always returns 0.

## ShortCondition

Checks conditions to open short position.

```
virtual int ShortCondition()
```

### Returned value

Then conditions are satisfied, it returns the value from 1 to 100 (depending on "strength" of a signal), if there isn't a signal to open short position, it returns 0.

### Note

The ShortCondition() method of a base class has no implementation of checking of conditions to open short position and always returns 0.



## Direction

Returns the value of "weighted" direction.

```
virtual double Direction()
```

### Returned value

It returns the value  $>0$  when upward direction (probably) and the returned value  $<0$  when downward direction. The absolute value depends on the "strength" of a signal.

### Note

If the filters are used, the result will depend on the filters.

## CExpertTrailing

CExpertTrailing is a base class for trailing algorithms, it does nothing but provides the interfaces.

How to use it:

1. Prepare an algorithm for trailing;
2. Create your own trailing class, inherited from CExpertTrailing class;
3. Override the virtual methods in your class with your own algorithms.

You can find an examples of trailing classes in the Expert\Trailing\ folder.

### Description

CExpertTrailing is a base class for implementation of trailing stop algoritms.

### Declaration

```
class CExpertTrailing : public CExpertBase
```

### Title

```
#include <Expert\ExpertTrailing.mqh>
```

### Class Methods

| Checking of Trailing Stop Conditions           |                                                              |
|------------------------------------------------|--------------------------------------------------------------|
| virtual <a href="#">CheckTrailingStopLong</a>  | Checks conditions to modify parameters of the long position  |
| virtual <a href="#">CheckTrailingStopShort</a> | Checks conditions to modify parameters of the short position |

## CheckTrailingStopLong

Checks conditions to modify parameters of the long position.

```
virtual bool CheckTrailingStopLong (
    CPositionInfo* position,    // pointer
    double& sl,                // Stop Loss
    double& tp                 // Take Profit
)
```

### Parameters

*position*

[in] Pointer to [CPositionInfo](#) class object.

*sl*

[in][out] Variable for Stop Loss price, passed by reference.

*tp*

[in][out] Variable for Take Profit price, passed by reference.

### Returned value

true if condition is satisfied, otherwise false.

### Note

The CheckTrailingStopLong(...) method of the base class always returns false.

## CheckTrailingStopShort

Checks conditions to modify parameters of the short position.

```
virtual bool CheckTrailingStopShort (
    CPositionInfo* position,    // pointer
    double& sl,                // Stop Loss
    double& tp                 // Take Profit
)
```

### Parameters

*position*

[in] Pointer to [CPositionInfo](#) class object.

*sl*

[in][out] Variable for Stop Loss price, passed by reference.

*tp*

[in][out] Variable for Take Profit price, passed by reference.

### Returned value

true if condition is satisfied, otherwise false.

### Note

The CheckTrailingStopShort(...) method of the base class always returns false.

## CExpertMoney

CExpertMoney is a base class for money and risk management algorithms.

### Description

CExpertMoney is a base class for implementation of money and risk management classes.

### Declaration

```
class CExpertMoney : public CObject
```

### Title

```
#include <Expert\ExpertMoney.mqh>
```

### Class Methods

| Access to Protected Data                   |                                                |
|--------------------------------------------|------------------------------------------------|
| <a href="#">Percent</a>                    | Sets the value of "Risk percent" parameter     |
| Initialization                             |                                                |
| virtual <a href="#">ValidationSettings</a> | Checks the settings                            |
| Checking of Trading Conditions             |                                                |
| virtual <a href="#">CheckOpenLong</a>      | Gets the volume for long position              |
| virtual <a href="#">CheckOpenShort</a>     | Gets the volume for short position             |
| virtual <a href="#">CheckReverse</a>       | Gets the volume for reverse of the position    |
| virtual <a href="#">CheckClose</a>         | Checks conditions to close the opened position |

## Percent

Sets the value of "Risk percent" parameter.

```
void Percent (
    double percent    // risk percent
)
```

### Parameters

*percent*

[in] Risk percent.

### Returned value

None.

## ValidationSettings

Checks the settings.

```
virtual bool ValidationSettings()
```

### Returned value

true if successful, otherwise false.

### Note

The ValidationSettings() method of the base class always returns true.

## CheckOpenLong

Gets the volume for long position.

```
virtual double CheckOpenLong(  
    double price,      // price  
    double sl          // Stop Loss  
)
```

### Parameters

*price*

[in] Opening price of long position.

*sl*

[in] Stop Loss price.

### Returned value

Trade volume for long position.



## CheckOpenShort

Gets the volume for short position.

```
virtual double CheckOpenShort(  
    double price,      // price  
    double sl          // Stop Loss  
)
```

### Parameters

*price*

[in] Opening price for short position.

*sl*

[in] Stop Loss price.

### Returned value

Trade volume for short position.

## CheckReverse

Gets the volume for reverse of the position.

```
virtual double CheckReverse(  
    CPositionInfo* position, // pointer  
    double sl               // Stop Loss  
)
```

### Parameters

*position*

[in] Pointer to [CPositionInfo](#) class object.

*sl*

[in] Stop Loss price.

### Returned value

Volume for reverse of the position.

## CheckClose

Checks conditions to close the opened position.

```
virtual double CheckClose()
```

### Returned value

true if condition is satisfied, otherwise false.

## Modules of Trade Signals

The standard delivery of the client terminal includes a set of ready-made modules of trade signals for "MQL5 Wizard". When creating an Expert Advisor in MQL5 Wizard, you can use any combination of the modules of trade signals (up to 64). The final decision on a trade operation is made on the basis of complex analysis of signals obtained from all included modules. The detailed description of the mechanism of making trade decisions is given [below](#).

The standard delivery includes the following modules of signals:

- [Signals of the Indicator Accelerator Oscillator](#)
- [Signals of the Indicator Adaptive Moving Average](#)
- [Signals of the Indicator Awesome Oscillator](#)
- [Signals of the Oscillator Bears Power](#)
- [Signals of the Oscillator Bulls Power](#)
- [Signals of the Oscillator Commodity Channel Index](#)
- [Signals of the Oscillator DeMarker](#)
- [Signals of the Indicator Double Exponential Moving Average](#)
- [Signals of the Indicator Envelopes](#)
- [Signals of the Indicator Fractal Adaptive Moving Average](#)
- [Signals of the Intraday Time Filter](#)
- [Signals of the Oscillator MACD](#)
- [Signals of the Indicator Moving Average](#)
- [Signals of the Indicator Parabolic SAR](#)
- [Signals of the Oscillator Relative Strength Index](#)
- [Signals of the Oscillator Relative Vigor Index](#)
- [Signals of the Oscillator Stochastic](#)
- [Signals of the Oscillator Triple Exponential Average](#)
- [Signals of the Indicator Triple Exponential Moving Average](#)
- [Signals of the Oscillator Williams Percent Range](#)

## The Mechanism of Making Trade Decisions on the Basis of Signal Modules

The mechanism of making trade decisions can be represented as the following list of basic principles:

- Each of the modules of signals has its set of market modules (certain combination of prices and values of an indicator).
- Each market model has a significance that may vary with the range of 1 to 100. The higher is the significance, the stronger the model is.
- Each of the models generates a forecast of direction of the price movement.
- A forecast of a module is the result of search for embedded models, and it is outputted as a number within the range of -100 to 100. The sign determines the direction of forecast movement (negative sign means the price will fall, positive sign means the price will rise). The absolute value

corresponds to the strength of the best found model.

- The forecast of each module is sent to the final "voting" with a weight coefficient of 0 to 1 specified in its settings ("Weight").
- The result of voting is a number within the range of -100 to 100, where the sign determines direction of the forecast movement, and the absolute value characterizes the strength of the signal. It is calculated as the arithmetical mean of weighted forecasts of all the modules of signals.

Each generated Expert Advisor has two adjustable settings – threshold levels of opening and closing a position (ThresholdOpen and ThresholdClose) that can be equal to a value in the range of 0 to 100. If the strength of final signal exceeds a threshold level, a trade operation that corresponds to the sign of the signal is performed.

## Examples

Consider an Expert Advisor with the following threshold levels: ThresholdOpen=20 and ThresholdClose=90. Two modules of signals participate in making decisions on trade operations: the [MA](#) module with weight 0.4 and the [Stochastic](#) module with weight 0.8. Let's analyze two variants of obtained trade signals:

### Variant 1.

The price crossed the rising MA upwards. This case corresponds to one of the market models implemented in the [MA module](#). This model implies a rise of price. Its significance is equal to 100. At the same time, the Stochastic oscillator turned down and formed a divergence with price. This case corresponds to one of the models implemented in the [Stochastic module](#). This model implies a fall of price. The weight of this model is 80.

Let's calculate the result of final "voting". The rate obtained from the MA module is calculated as  $0.4 * 100 = 40$ . The value from the Stochastic module is calculated as  $0.8 * (-80) = -64$ . The final value is calculated as the arithmetical mean of these two rates:  $(40 - 64) / 2 = -12$ . The result of voting is the signal for selling with relative strength equal to 12. The threshold level that is equal to 20 is not reached. Thus a trade operation is not performed.

### Variant 2.

The price crossed the rising MA downwards. This case corresponds to one of the models implemented in the [MA module](#). This model implies a rise of price. Its significance is equal to 10. At the same time, the Stochastic oscillator turned down and formed a divergence with price. This case corresponds to one of the models implemented in the [Stochastic module](#). This model implies a fall of price. The weight of this model is 80.

Let's calculate the result of final "voting". The rate obtained from the MA module is calculated as  $0.4 * 10 = 4$ . The value from the Stochastic module is calculated as  $0.8 * (-80) = -64$ . The final value is calculated as the arithmetical mean of these two rates:  $(4 - 64) / 2 = -30$ . The result of voting is the signal for selling with relative strength equal to 30. The threshold level that is equal to 20 is reached. Thus the result is the signal for opening a short position.



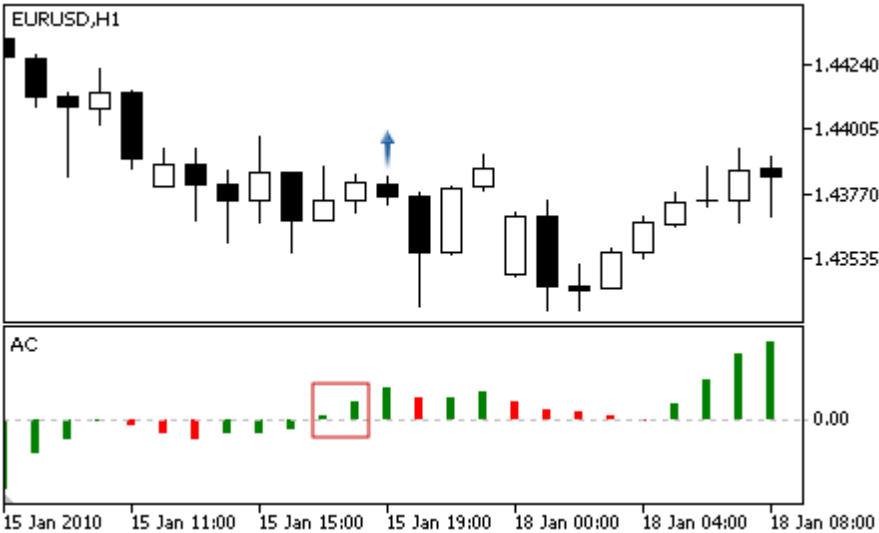
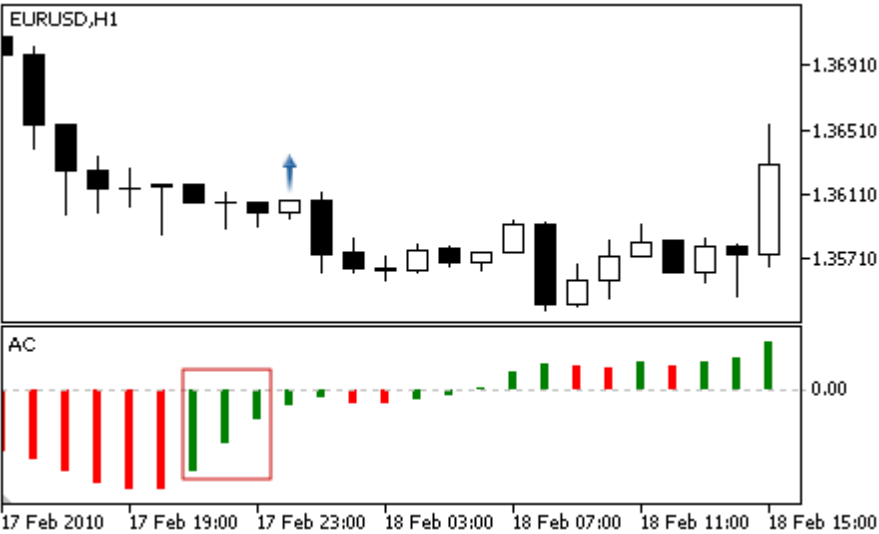
- a) Divergence of the price and the Stochastic oscillator (variants 1 and 2).
- b) The price crossed the MA indicator upwards (variant 1).
- c) The price crossed the MA indicator downwards (variants 2).

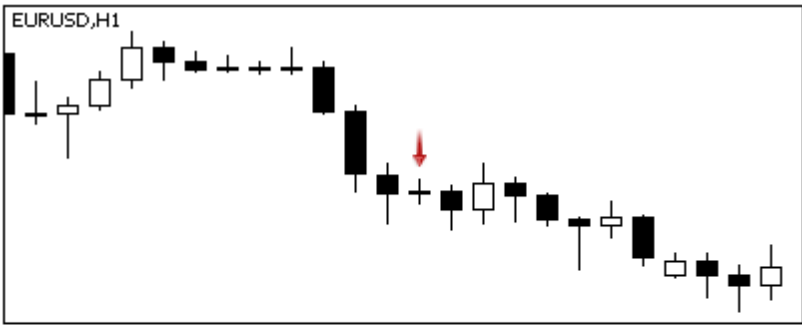
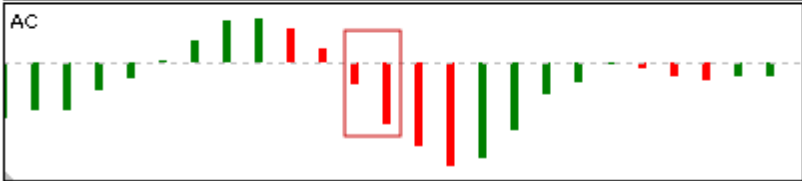
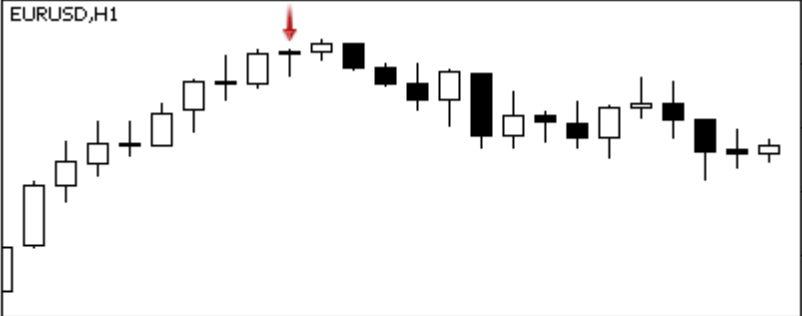
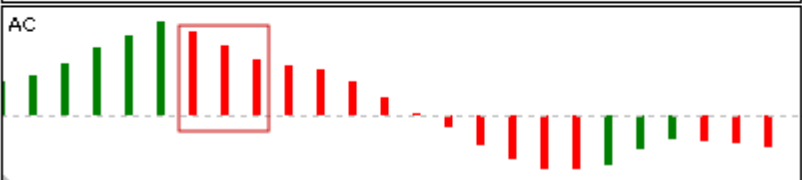
## Signals of the Indicator Accelerator Oscillator

This module is based on the market models of the indicator [Accelerator Oscillator](#). The mechanism of making trade decisions based on signals obtained from the modules is described in a [separate section](#).

### Conditions of Generation of Signals

Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

| Signal Type | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| For buying  | <ul style="list-style-type: none"> <li>The indicator value is above 0 and it rises at the analyzed and at the previous bars.</li> </ul>  <ul style="list-style-type: none"> <li>The indicator value is below 0 and it rises at the analyzed and at the previous bars.</li> </ul>  |
| For selling | <ul style="list-style-type: none"> <li>The indicator value is below 0 and it falls at the analyzed and at the previous bars.</li> </ul>                                                                                                                                                                                                                                                                                                                  |

| Signal Type              | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                          | <div><div><div><div>EURUSD,H1</div><div>1.45000<br/>1.44630<br/>1.44260<br/>1.43890</div></div><div><div>AC</div><div>0.00</div></div><div>14 Jan 2010 14 Jan 19:00 14 Jan 23:00 15 Jan 03:00 15 Jan 07:00 15 Jan 11:00 15 Jan 15:00</div></div><div><ul style="list-style-type: none"><li>• The indicator value is below 0 and it falls at the analyzed and at the previous bars.</li></ul></div><div><div><div>EURUSD,H1</div><div>1.36400<br/>1.36040<br/>1.35680<br/>1.35320</div></div><div><div>AC</div><div>0.00</div></div><div>19 Feb 2010 19 Feb 21:00 22 Feb 02:00 22 Feb 06:00 22 Feb 10:00 22 Feb 14:00 22 Feb 18:00</div></div></div> |
| No objections to buying  | The indicator value grows at the analyzed bar.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| No objections to selling | The indicator value falls at the analyzed bar.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

Note

Depending on the mode of operation of an Expert Advisor ("Every tick" or "Open prices only") an analyzed bar is either the current bar (with index 0), or the last formed bar (with index 1).

Adjustable Parameters

This module has the following adjustable parameters:



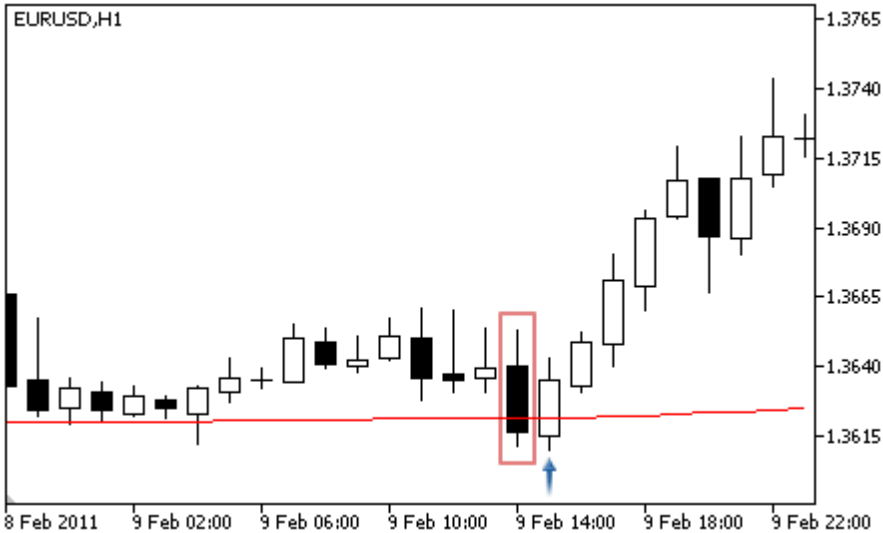
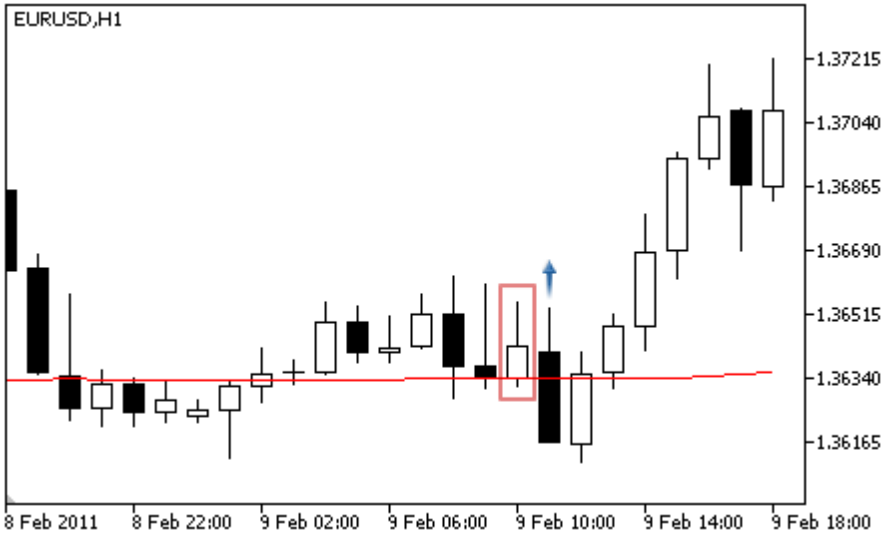
| Parameter | Description                                            |
|-----------|--------------------------------------------------------|
| Weight    | Weight of signal of the module in the interval 0 to 1. |

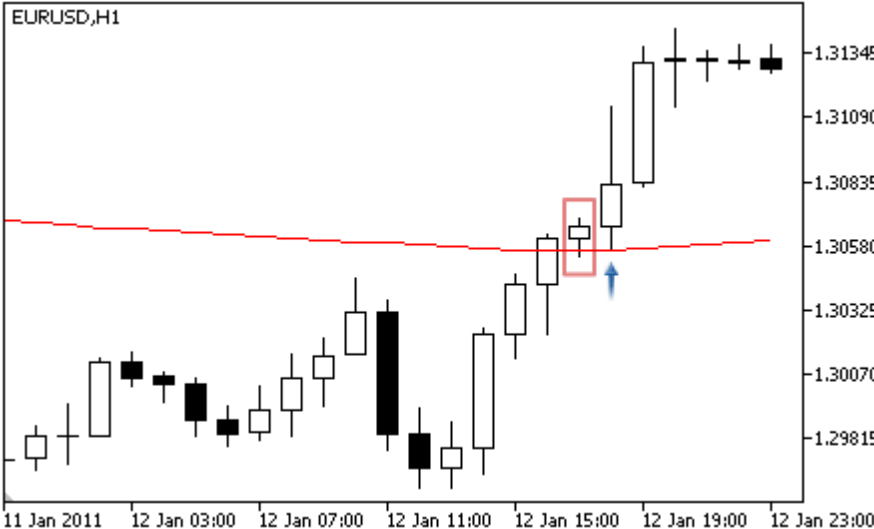
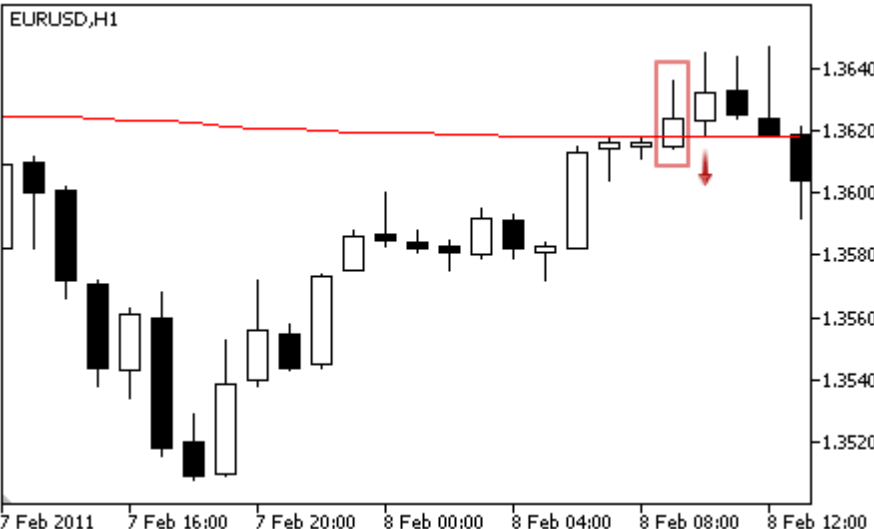
## Signals of the Indicator Adaptive Moving Average

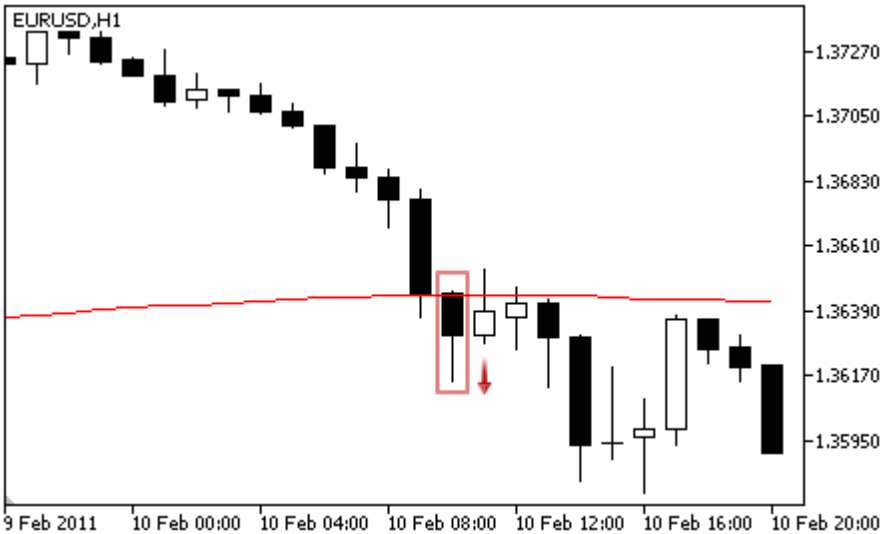
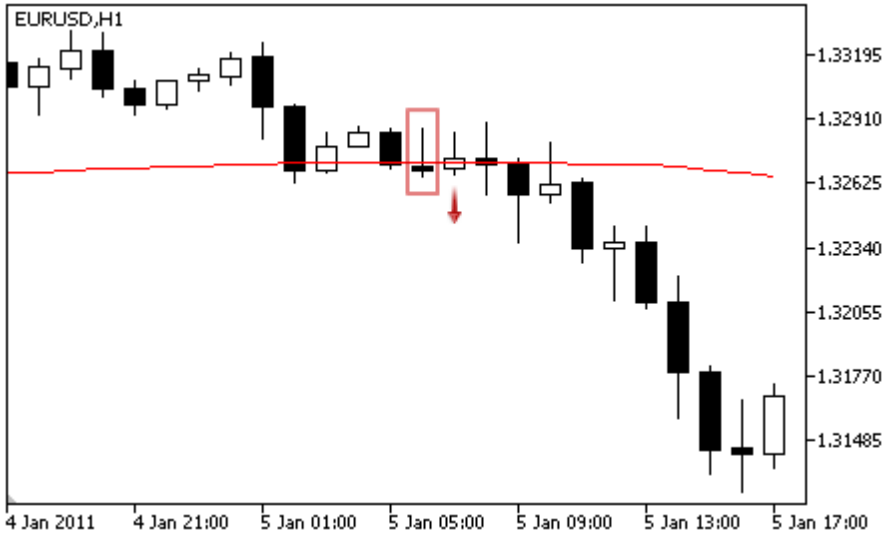
This module is based on the market models of the indicator [Adaptive Moving Average](#). The mechanism of making trade decisions based on signals obtained from the modules is described in a [separate section](#).

### Conditions of Generation of Signals

Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

| Signal Type | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| For buying  | <ul style="list-style-type: none"> <li>The price has crossed the indicator downwards (the Open price of the analyzed bar is above the indicator and the Close price is below the indicator) and the indicator rises (weak signal).</li> </ul>  <ul style="list-style-type: none"> <li><b>Moving Average crossover.</b> The price has crossed the indicator upwards (the Open price of the analyzed bar is below the indicator and the Close price is above the indicator) and the indicator rises (strong signal).</li> </ul>  |

| Signal Type | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|             | <ul style="list-style-type: none"> <li>The lower shadow of the bar has crossed the indicator (the Open and Close prices of the analyzed bar is above the indicator, and the Low price is below the indicator) and the indicator rises (weak signal).</li> </ul>                                                                                                                                                                                                                                                                 |
| For selling | <ul style="list-style-type: none"> <li>The price has crossed the indicator upwards (the Open price of the analyzed bar is below the indicator and the Close price is above the indicator) and the indicator falls (weak signal).</li> </ul>  <ul style="list-style-type: none"> <li><b>Moving Average crossover.</b> The price has crossed the indicator downwards (the Open price of the analyzed bar is above the indicator and the Close price is below the indicator) and the indicator falls (strong signal).</li> </ul> |

| Signal Type              | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                          | <div></div> <ul style="list-style-type: none"><li>The upper shadow of the bar has crossed the indicator (the Open and Close prices of the analyzed bar is below the indicator, and the High price is above the indicator) and the indicator falls (weak signal).</li></ul> <div></div> |
| No objections to buying  | The prices is above the indicator.                                                                                                                                                                                                                                                                                                                                                                                                                          |
| No objections to selling | The price is below the indicator.                                                                                                                                                                                                                                                                                                                                                                                                                           |

Note

Depending on the mode of operation of an Expert Advisor ("Every tick" or "Open prices only") an analyzed bar is either the current bar (with index 0), or the last formed bar (with index 1).

Adjustable Parameters

This module has the following adjustable parameters:

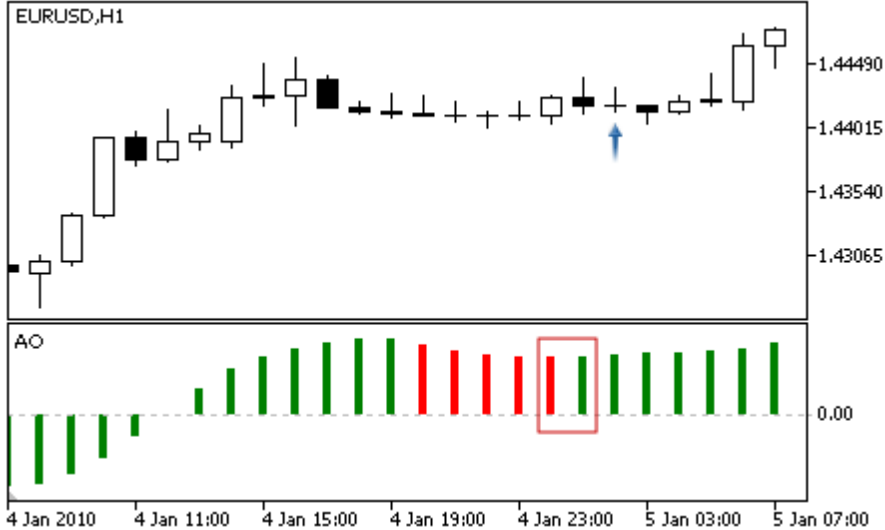
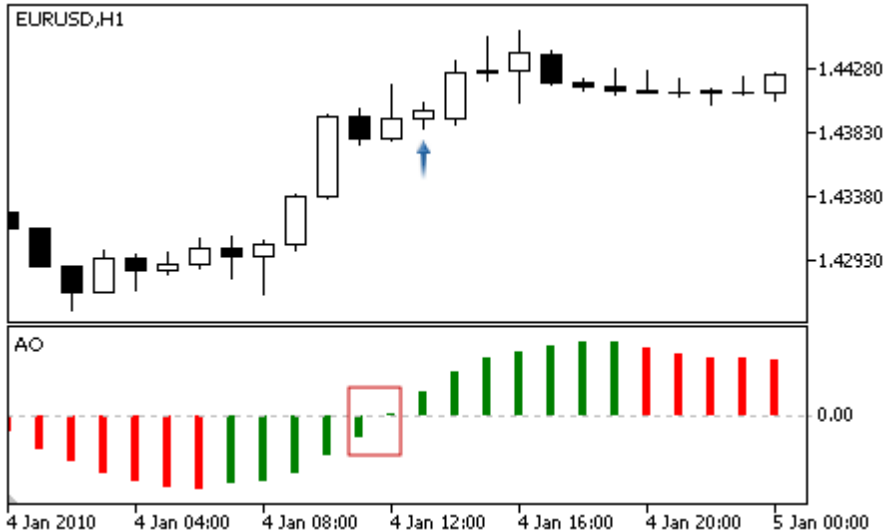
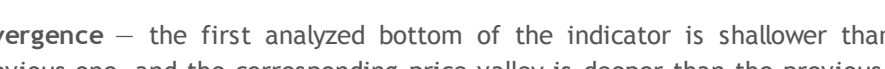
| Parameter | Description                                                           |
|-----------|-----------------------------------------------------------------------|
| Weight    | Weight of signal of the module in the interval 0 to 1.                |
| PeriodMA  | Period of averaging of the indicator.                                 |
| Shift     | Shift of the indicator along the time axis (in bars).                 |
| Method    | <a href="#">Method of averaging</a> .                                 |
| Applied   | A <a href="#">price series</a> used for calculation of the indicator. |

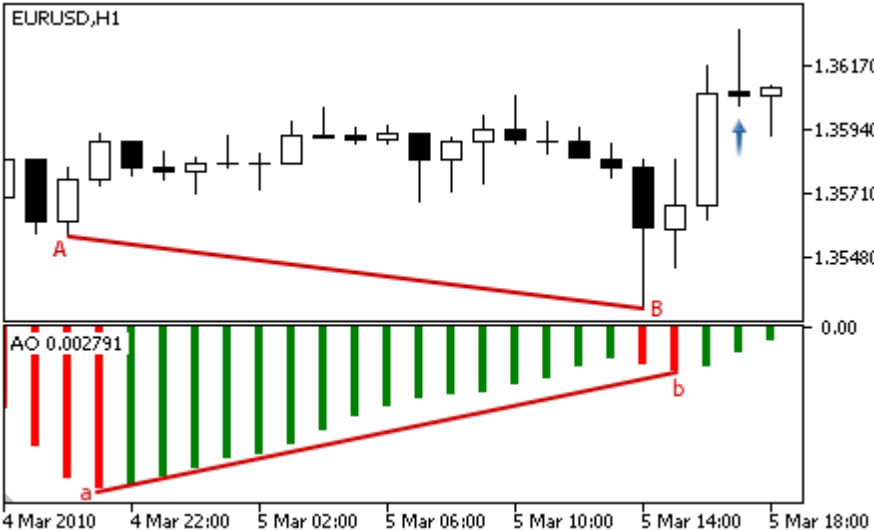
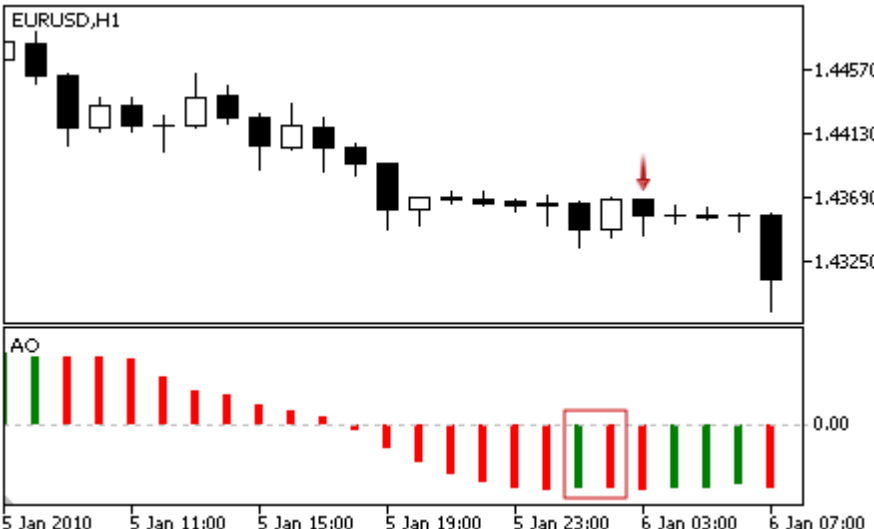
## Signals of the Indicator Awesome Oscillator

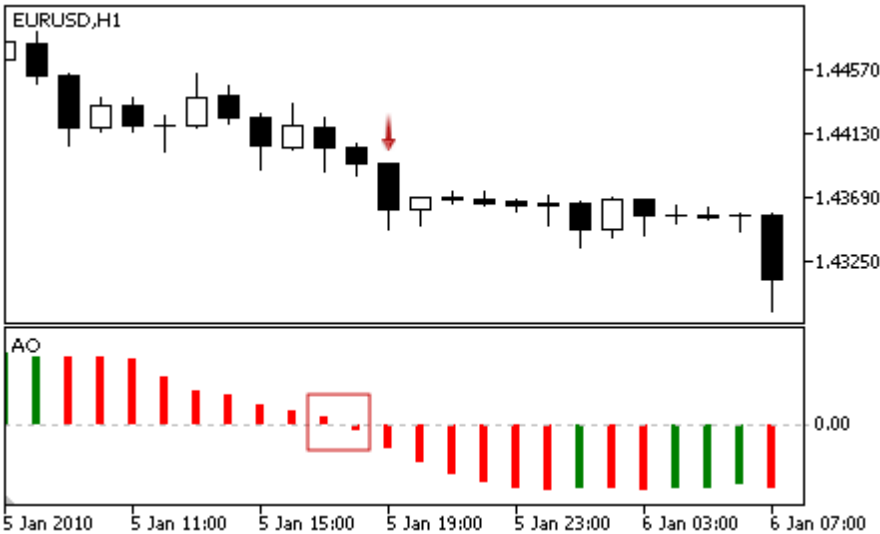
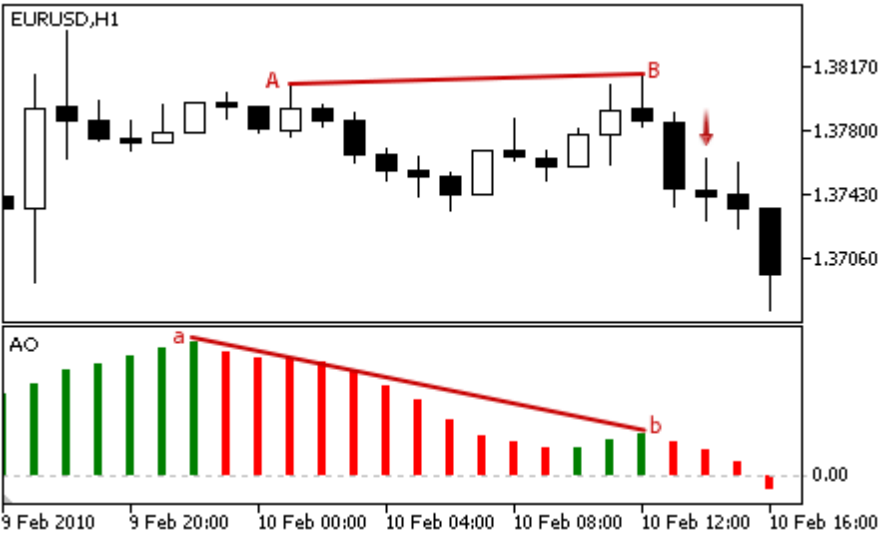
This module of signals is based on the market models of the indicator [Awesome Oscillator](#). The mechanism of making trade decisions based on signals obtained from the modules is described in a [separate section](#).

### Conditions of Generation of Signals

Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

| Signal Type | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| For buying  | <ul style="list-style-type: none"> <li> <b>Saucer</b> — value of the indicator at the analyzed bar rises, and it fell at the previous bars; at that, both values are above 0.            </li> <li> <b>Crossing the zero line</b> — value of the indicator is above 0 at the analyzed bar, and it is below 0 at the previous bar.            </li> <li> <b>Divergence</b> — the first analyzed bottom of the indicator is shallower than the previous one, and the corresponding price valley is deeper than the previous one.            </li> </ul> |

| Signal Type | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|             | <p>In addition, the indicator must not rise above the zero level.</p>                                                                                                                                                                                                                                                                                                         |
| For selling | <ul style="list-style-type: none"> <li>• <b>Saucer</b> – value of the indicator at the analyzed bar falls, and it rose at the previous bars; at that, both values are below 0.</li> </ul>  <ul style="list-style-type: none"> <li>• <b>Crossing the zero line</b> – value of the indicator is below 0 at the analyzed bar, and it is above 0 at the previous bar.</li> </ul> |

| Signal Type              | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                                                |
|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                          |  <ul style="list-style-type: none"> <li>• <b>Divergence</b> – the first analyzed peak of the indicator is lower than the previous one, and the corresponding price peak is higher than the previous one. In addition, the indicator must not fall below the zero level.</li> </ul>  |
| No objections to buying  | The indicator value grows at the analyzed bar.                                                                                                                                                                                                                                                                                                                                                                                                           |
| No objections to selling | The indicator value falls at the analyzed bar.                                                                                                                                                                                                                                                                                                                                                                                                           |

#### Note

Depending on the mode of operation of an Expert Advisor ("Every tick" or "Open prices only") an analyzed bar is either the current bar (with index 0), or the last formed bar (with index 1).

## Adjustable Parameters



This module has the following adjustable parameters:

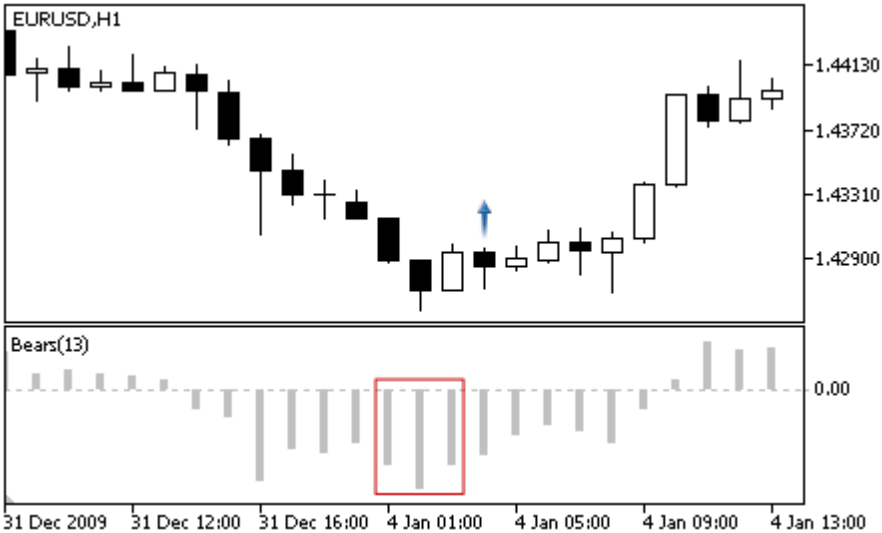
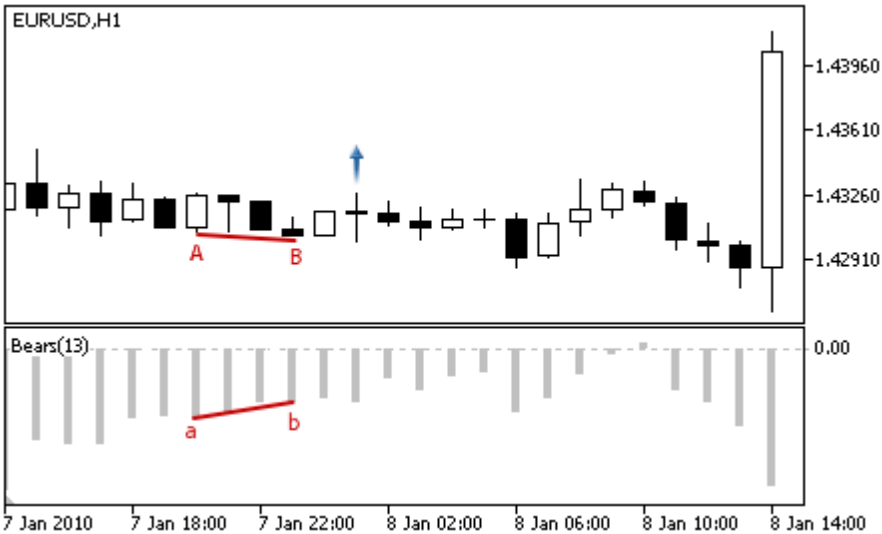
| Parameter | Description                                            |
|-----------|--------------------------------------------------------|
| Weight    | Weight of signal of the module in the interval 0 to 1. |

## Signals of the Oscillator Bears Power

This module of signals is based on the market models of the oscillator [Bears Power](#). The mechanism of making trade decisions based on signals obtained from the modules is described in a [separate section](#).

### Conditions of Generation of Signals

Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

| Signal Type | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| For buying  | <ul style="list-style-type: none"> <li> <b>Reverse</b> — the oscillator turned upwards and its value at the analyzed bar is below 0.            </li> <li> <b>Divergence</b> — the first analyzed bottom of the oscillator is higher than the previous one, and the corresponding price bottom is lower than the previous one. In addition, the oscillator must not rise above the zero level.            </li> </ul> |
| For selling | No signals for selling.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |

| Signal Type              | Description of Conditions               |
|--------------------------|-----------------------------------------|
| No objections to buying  | Value of the oscillator is less than 0. |
| No objections to selling | No signals.                             |

**Note**

Depending on the mode of operation of an Expert Advisor ("Every tick" or "Open prices only") an analyzed bar is either the current bar (with index 0), or the last formed bar (with index 1).

## Adjustable Parameters

This module has the following adjustable parameters:

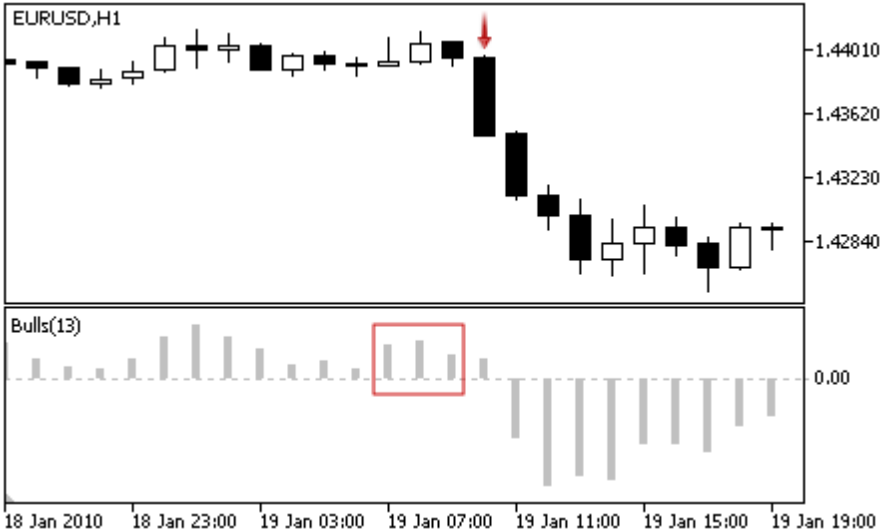
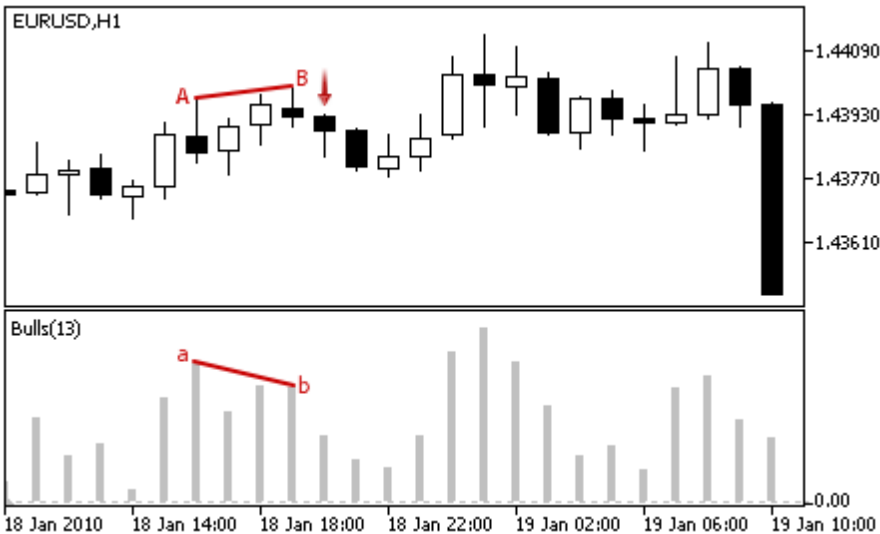
| Parameter   | Description                                            |
|-------------|--------------------------------------------------------|
| Weight      | Weight of signal of the module in the interval 0 to 1. |
| PeriodBears | Period of calculation of the oscillator.               |

## Signals of the Oscillator Bulls Power

This module of signals is based on the market models of the oscillator [Bulls Power](#). The mechanism of making trade decisions based on signals obtained from the modules is described in a [separate section](#).

### Conditions of Generation of Signals

Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

| Signal Type | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| For buying  | No signals for buying.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| For selling | <ul style="list-style-type: none"> <li> <b>Reverse</b> – the oscillator turned downwards and its value at the analyzed bar is above 0.            </li> <li> <b>Divergence</b> – the first analyzed peak of the oscillator is lower than the previous one, and the corresponding price peak is higher than the previous peak. In addition, the oscillator must not fall below the zero level.            </li> </ul> |

| Signal Type              | Description of Conditions                  |
|--------------------------|--------------------------------------------|
| No objections to buying  | No signals.                                |
| No objections to selling | Value of the oscillator is greater than 0. |

**Note**

Depending on the mode of operation of an Expert Advisor ("Every tick" or "Open prices only") an analyzed bar is either the current bar (with index 0), or the last formed bar (with index 1).

## Adjustable Parameters

This module has the following adjustable parameters:

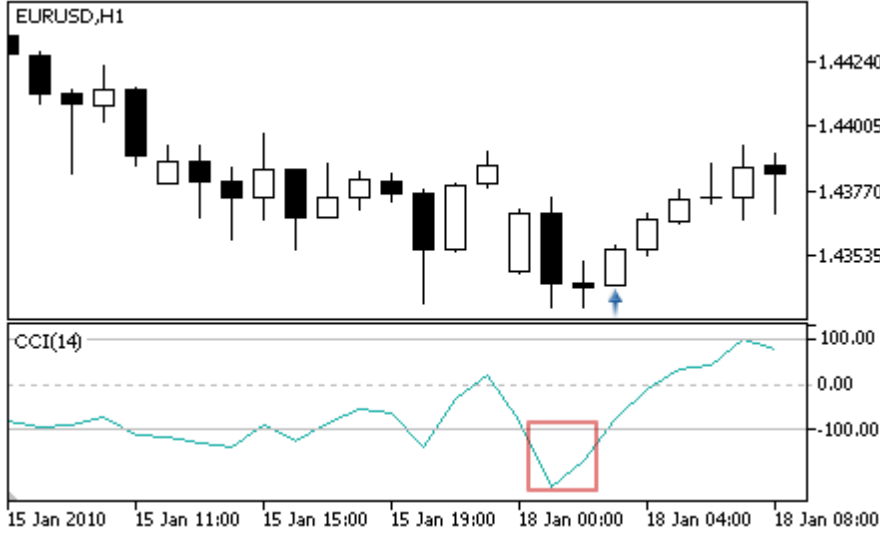
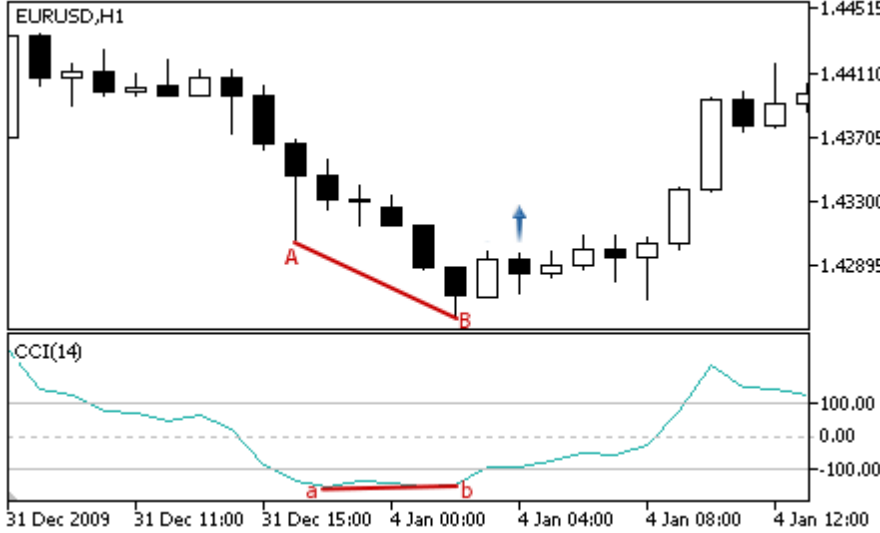
| Parameter   | Description                                            |
|-------------|--------------------------------------------------------|
| Weight      | Weight of signal of the module in the interval 0 to 1. |
| PeriodBulls | Period of calculation of the oscillator.               |

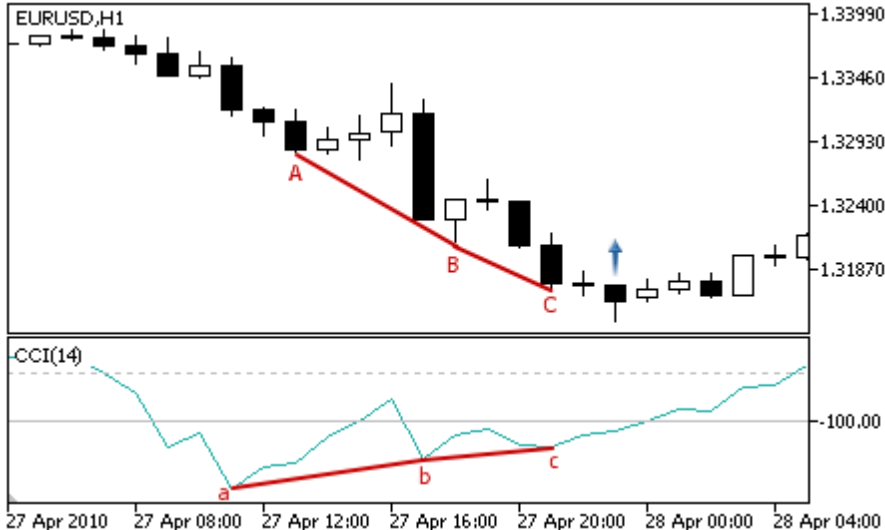
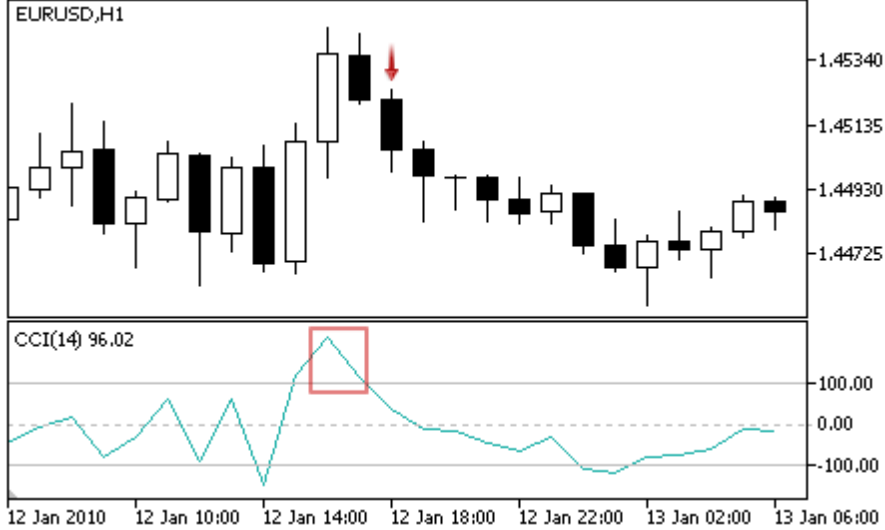
## Signals of the Oscillator Commodity Channel Index

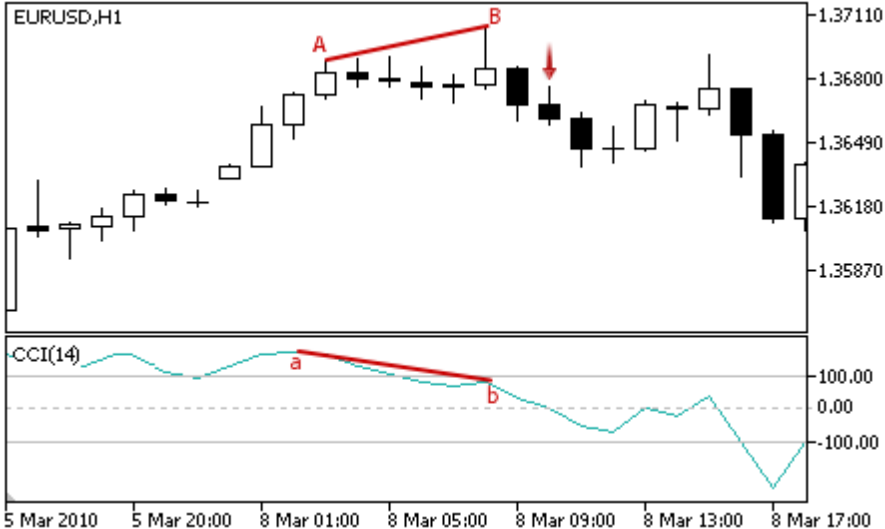
This module of signals is based on the market models of the oscillator [Commodity Channel Index](#). The mechanism of making trade decisions based on signals obtained from the modules is described in a [separate section](#).

### Conditions of Generation of Signals

Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

| Signal Type | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| For buying  | <ul style="list-style-type: none"> <li> <b>Reverse behind the level of overselling</b> – the oscillator turned upwards and its value at the analyzed bar is behind the level of overselling (default value is -100).            </li> <li> <b>Divergence</b> – the first analyzed bottom of the oscillator is higher than the previous one, and the corresponding price bottom is lower than the previous one.            </li> <li> <b>Double divergence</b> – the oscillator form three consequent bottoms, each of them is higher than the previous one; and the price formed three corresponding         </li> </ul> |

| Signal Type | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|             | <p>bottoms, and each of them is lower than the previous one.</p>                                                                                                                                                                                                                                                                                                                                                                                 |
| For selling | <ul style="list-style-type: none"> <li>• <b>Reverse behind the overbought level</b> – the oscillator turned downwards and its value at the analyzed bar is behind the overbought level (default value is 100).</li> </ul>  <ul style="list-style-type: none"> <li>• <b>Divergence</b> – the first analyzed peak of the oscillator is lower than the previous one, and the corresponding price peak is higher than the previous peak.</li> </ul> |

| Signal Type              | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                           |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                          |  <p>• <b>Double divergence</b> – the oscillator formed three consequent peaks, each of them is lower than the previous one; and the price formed three corresponding peaks, each of them is higher than the previous one.</p>  |
| No objections to buying  | Value of the oscillator grows at the analyzed bar.                                                                                                                                                                                                                                                                                                                                                  |
| No objections to selling | Value of the oscillator falls at the analyzed bar.                                                                                                                                                                                                                                                                                                                                                  |

#### Note

Depending on the mode of operation of an Expert Advisor ("Every tick" or "Open prices only") an analyzed bar is either the current bar (with index 0), or the last formed bar (with index 1).

## Adjustable Parameters



This module has the following adjustable parameters:

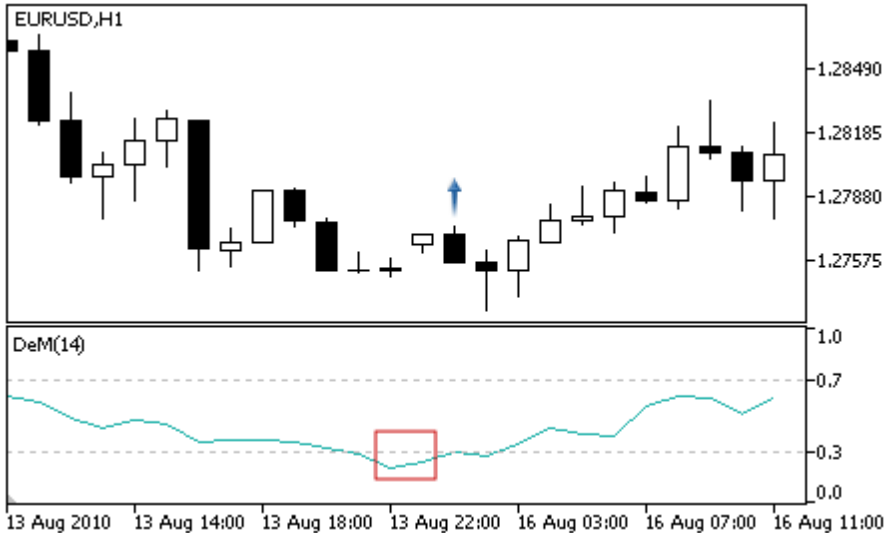
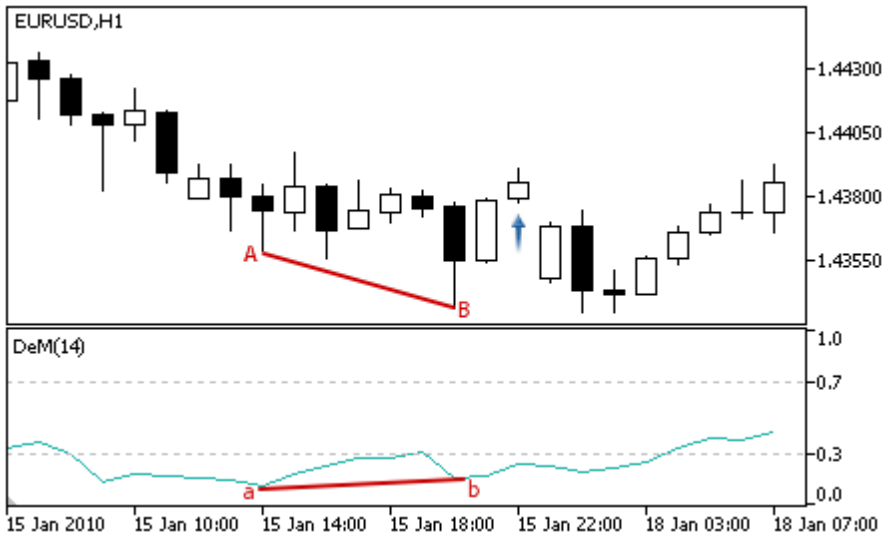
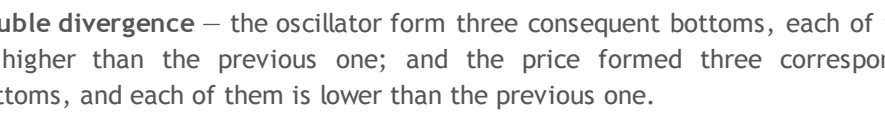
| Parameter | Description                                                            |
|-----------|------------------------------------------------------------------------|
| Weight    | Weight of signal of the module in the interval 0 to 1.                 |
| PeriodCCI | Period of calculation of the oscillator.                               |
| Applied   | A <a href="#">price series</a> used for calculation of the oscillator. |

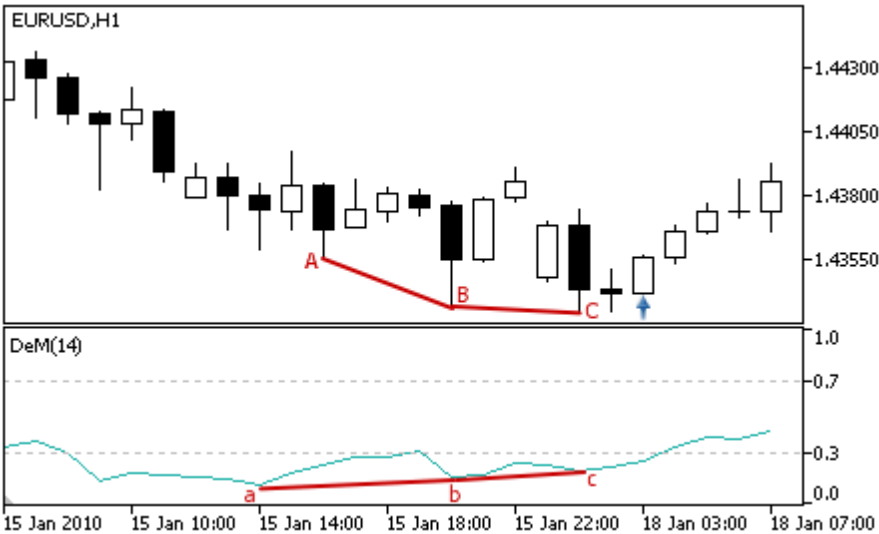
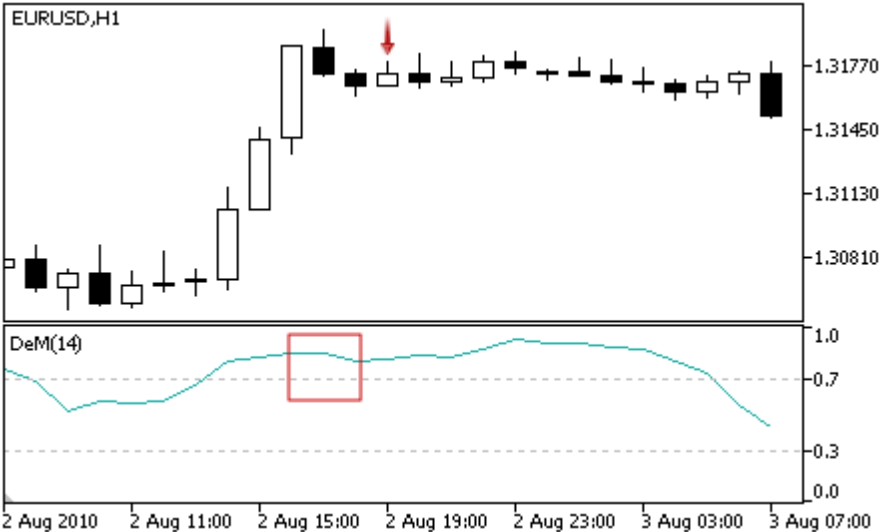
## Signals of the Oscillator DeMarker

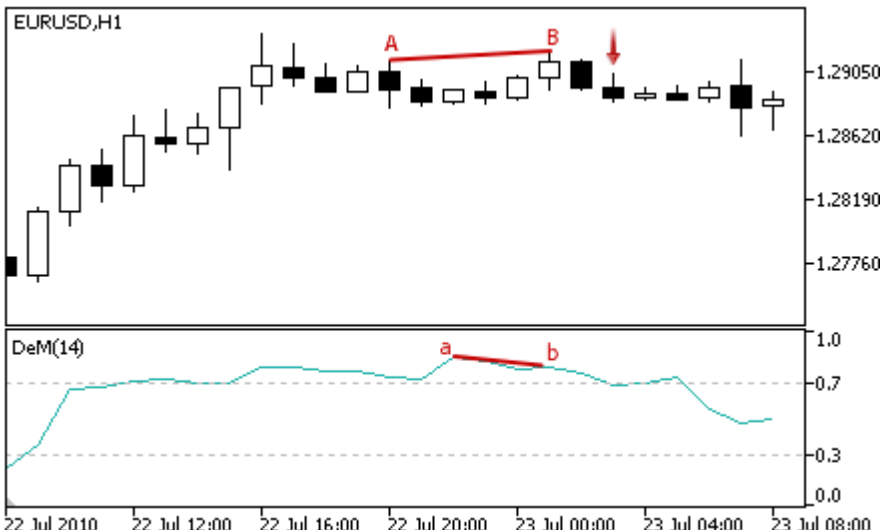
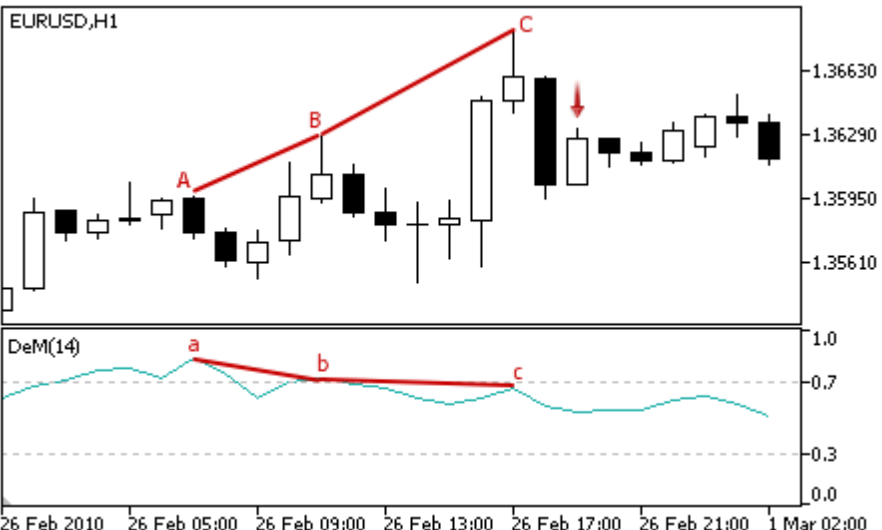
This module of signals is based on the market models of the oscillator [DeMarker](#). The mechanism of making trade decisions based on signals obtained from the modules is described in a [separate section](#).

### Conditions of Generation of Signals

Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

| Signal Type | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| For buying  | <ul style="list-style-type: none"> <li> <b>Reverse behind the oversold level</b> – the oscillator turned upwards and its value at the analyzed bar is behind the oversold level (default value is 0.3).            </li> <li> <b>Divergence</b> – the first analyzed bottom of the oscillator is higher than the previous one, and the corresponding price bottom is lower than the previous one.            </li> <li> <b>Double divergence</b> – the oscillator form three consequent bottoms, each of them is higher than the previous one; and the price formed three corresponding bottoms, and each of them is lower than the previous one.            </li> </ul> |

| Signal Type | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|             |  <p>The top chart is a candlestick plot for EURUSD on an H1 timeframe. It shows a series of price bars from 15 Jan 10:00 to 18 Jan 07:00. A red line connects points A, B, and C, indicating a downward trend. The bottom chart is the DeM(14) oscillator, showing a green line with points a, b, and c marked, corresponding to the price points above. The oscillator's value decreases as the price moves down.</p>                                                                                                                                                                                                                                                                                                                                                                                   |
| For selling | <ul style="list-style-type: none"> <li>• <b>Reverse behind the overbought level</b> – the oscillator turned downwards and its value at the analyzed bar is behind the overbought level (default value is 0.7).</li> </ul>  <p>The top chart is a candlestick plot for EURUSD on an H1 timeframe. It shows a series of price bars from 2 Aug 10:00 to 3 Aug 07:00. A red arrow points to a peak in the price. The bottom chart is the DeM(14) oscillator, showing a green line with a peak highlighted by a red box. The oscillator's value is above the 0.7 level at the peak.</p> <ul style="list-style-type: none"> <li>• <b>Divergence</b> – the first analyzed peak of the oscillator is lower than the previous one, and the corresponding price peak is higher than the previous peak.</li> </ul> |

| Signal Type              | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                           |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                          |  <p>• <b>Double divergence</b> – the oscillator formed three consequent peaks, each of them is lower than the previous one; and the price formed three corresponding peaks, each of them is higher than the previous one.</p>  |
| No objections to buying  | Value of the oscillator grows at the analyzed bar.                                                                                                                                                                                                                                                                                                                                                  |
| No objections to selling | Value of the oscillator falls at the analyzed bar.                                                                                                                                                                                                                                                                                                                                                  |

#### Note

Depending on the mode of operation of an Expert Advisor ("Every tick" or "Open prices only") an analyzed bar is either the current bar (with index 0), or the last formed bar (with index 1).

## Adjustable Parameters

This module has the following adjustable parameters:

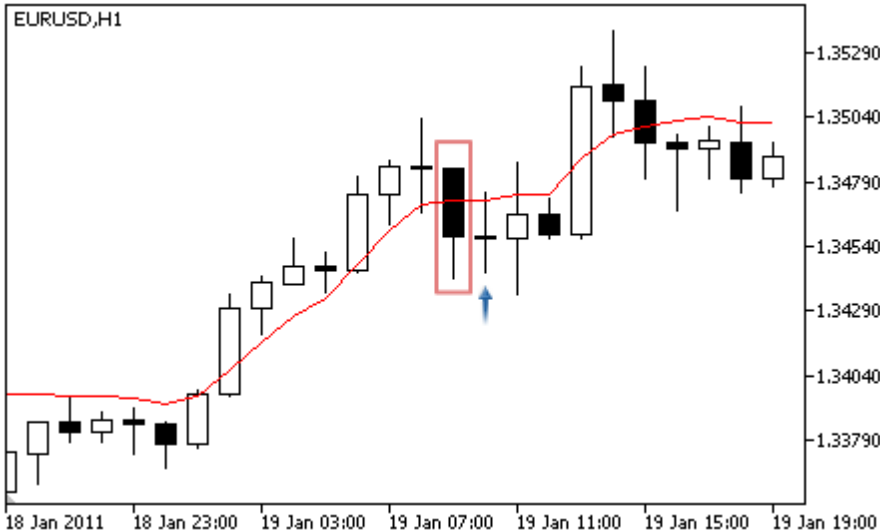
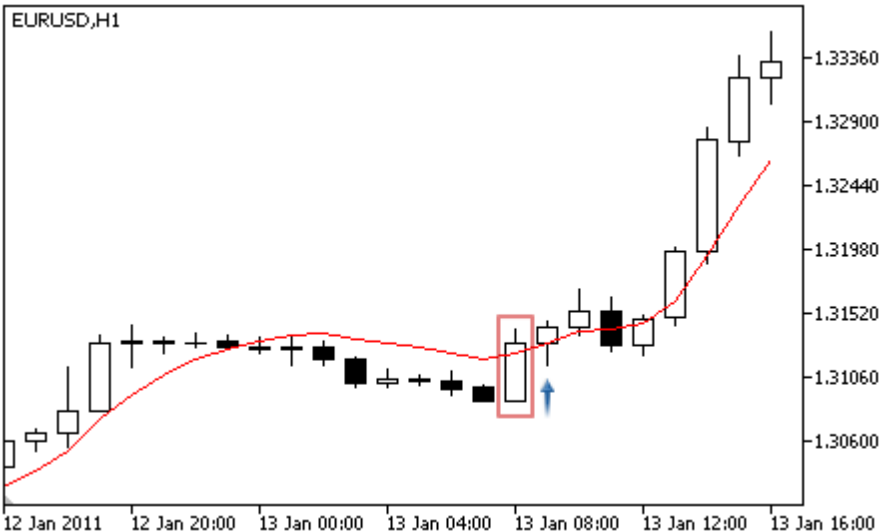
| Parameter | Description                                            |
|-----------|--------------------------------------------------------|
| Weight    | Weight of signal of the module in the interval 0 to 1. |
| PeriodDeM | Period of calculation of the oscillator.               |

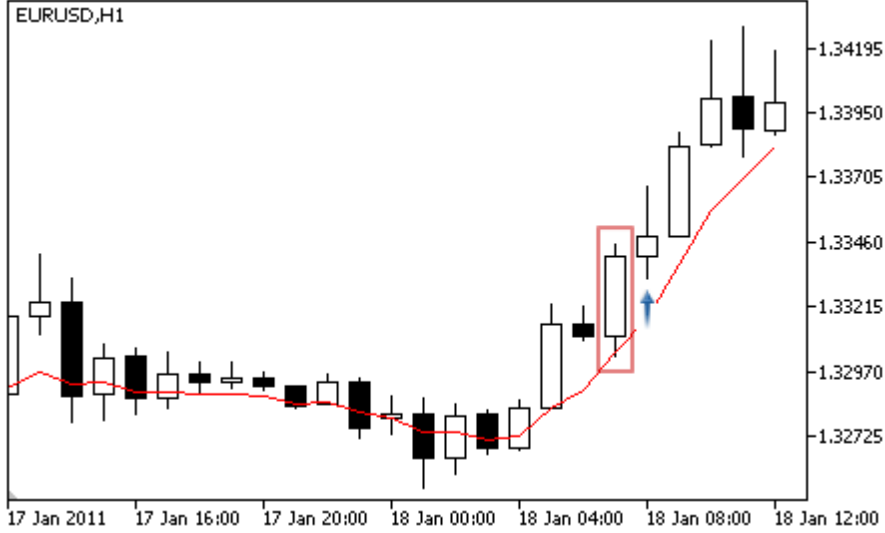
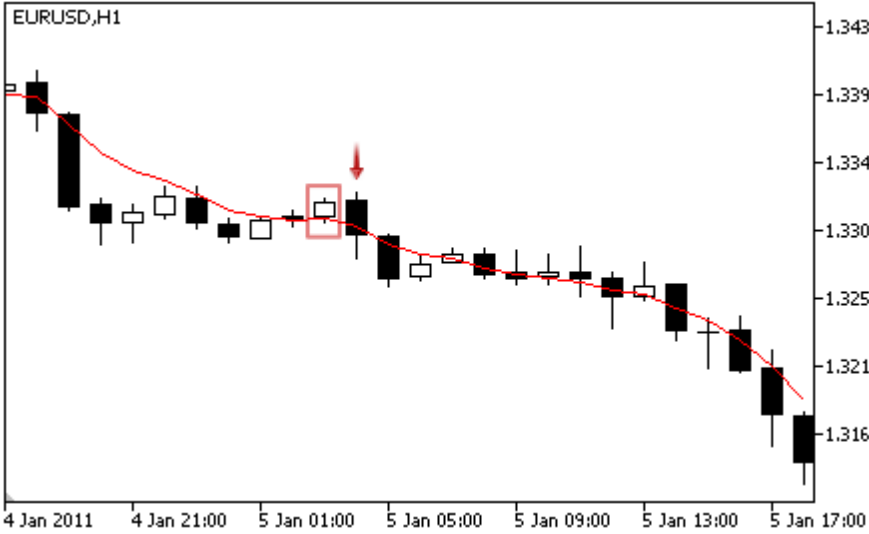
## Signals of the Indicator Double Exponential Moving Average

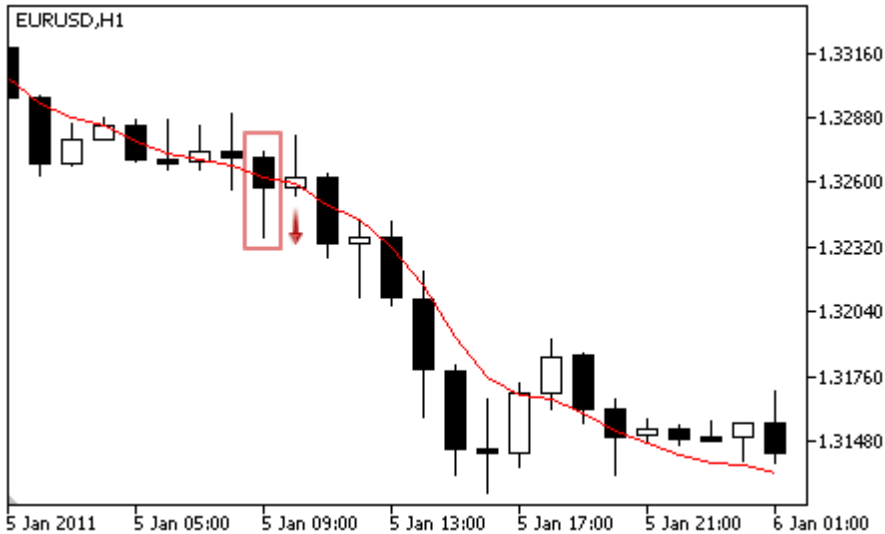
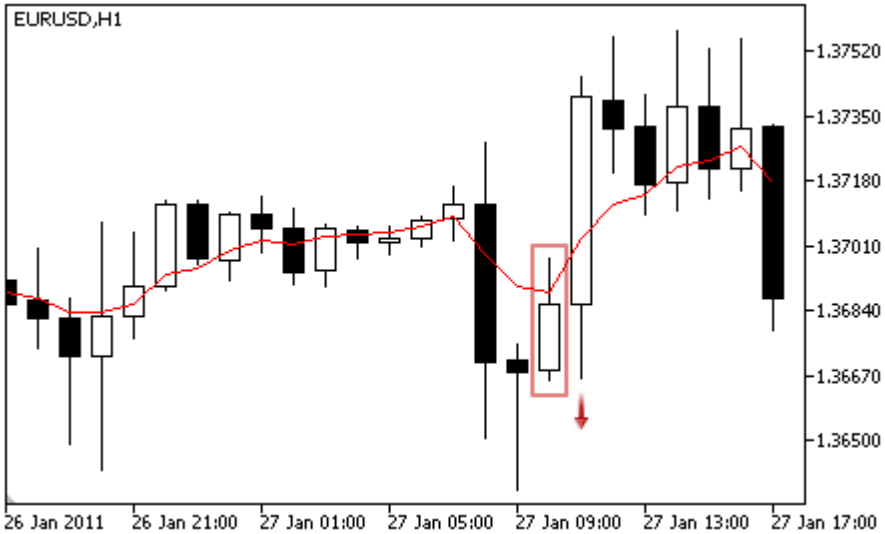
This module is based on the market models of the indicator [Double Exponential Moving Average](#). The mechanism of making trade decisions based on signals obtained from the modules is described in a [separate section](#).

### Conditions of Generation of Signals

Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

| Signal Type | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| For buying  | <ul style="list-style-type: none"> <li>The price has crossed the indicator downwards (the Open price of the analyzed bar is above the indicator and the Close price is below the indicator) and the indicator rises (weak signal).</li> </ul>  <ul style="list-style-type: none"> <li><b>Moving Average crossover.</b> The price has crossed the indicator upwards (the Open price of the analyzed bar is below the indicator and the Close price is above the indicator) and the indicator rises (strong signal).</li> </ul>  |

| Signal Type | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|             | <ul style="list-style-type: none"> <li>The lower shadow of the bar has crossed the indicator (the Open and Close prices of the analyzed bar is above the indicator, and the Low price is below the indicator) and the indicator rises (weak signal).</li> </ul>                                                                                                                                                                                                                                                                 |
| For selling | <ul style="list-style-type: none"> <li>The price has crossed the indicator upwards (the Open price of the analyzed bar is below the indicator and the Close price is above the indicator) and the indicator falls (weak signal).</li> </ul>  <ul style="list-style-type: none"> <li><b>Moving Average crossover.</b> The price has crossed the indicator downwards (the Open price of the analyzed bar is above the indicator and the Close price is below the indicator) and the indicator falls (strong signal).</li> </ul> |

| Signal Type              | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                               |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                          |  <ul style="list-style-type: none"> <li>The upper shadow of the bar has crossed the indicator (the Open and Close prices of the analyzed bar is below the indicator, and the High price is above the indicator) and the indicator falls (weak signal).</li> </ul>  |
| No objections to buying  | The price is above the indicator.                                                                                                                                                                                                                                                                                                                                                                                                       |
| No objections to selling | The price is below the indicator.                                                                                                                                                                                                                                                                                                                                                                                                       |

#### Note

Depending on the mode of operation of an Expert Advisor ("Every tick" or "Open prices only") an analyzed bar is either the current bar (with index 0), or the last formed bar (with index 1).

## Adjustable Parameters



This module has the following adjustable parameters:

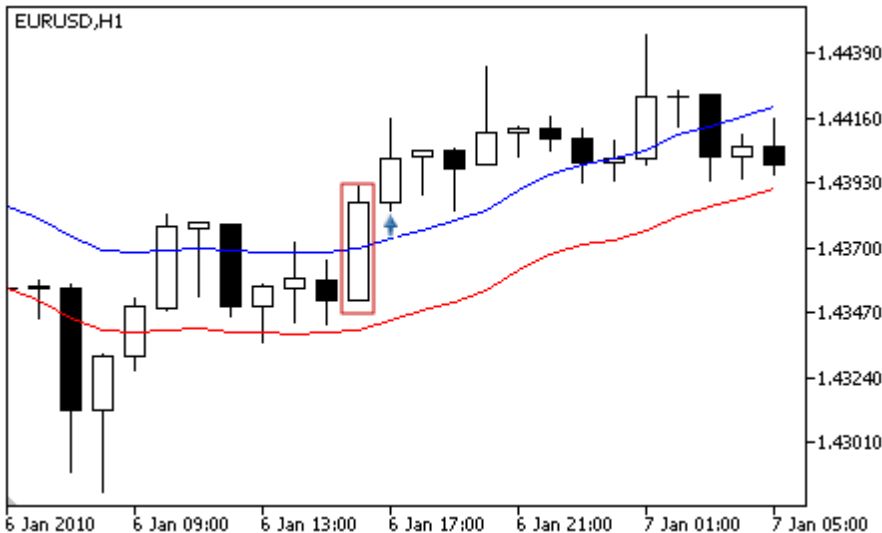
| Parameter | Description                                                           |
|-----------|-----------------------------------------------------------------------|
| Weight    | Weight of signal of the module in the interval 0 to 1.                |
| PeriodMA  | Period of averaging of the indicator.                                 |
| Shift     | Shift of the indicator along the time axis (in bars).                 |
| Method    | <a href="#">Method of averaging</a> .                                 |
| Applied   | A <a href="#">price series</a> used for calculation of the indicator. |

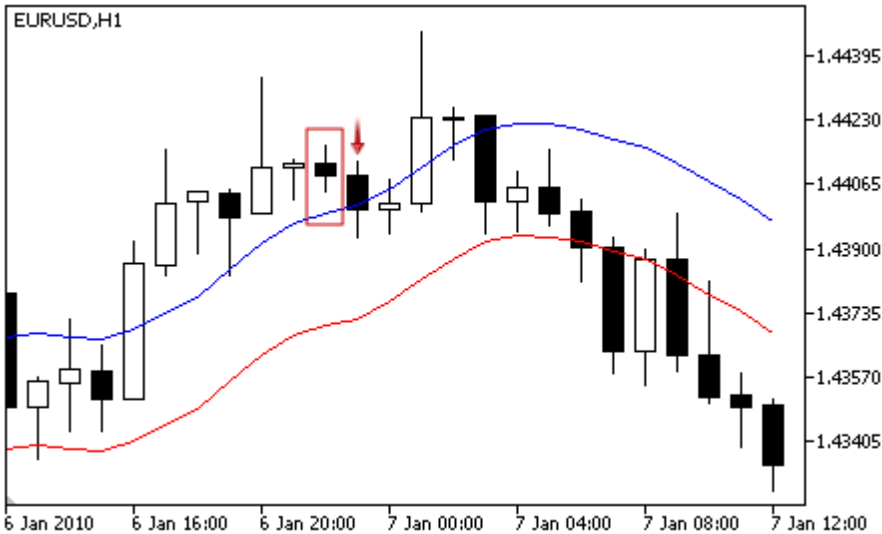
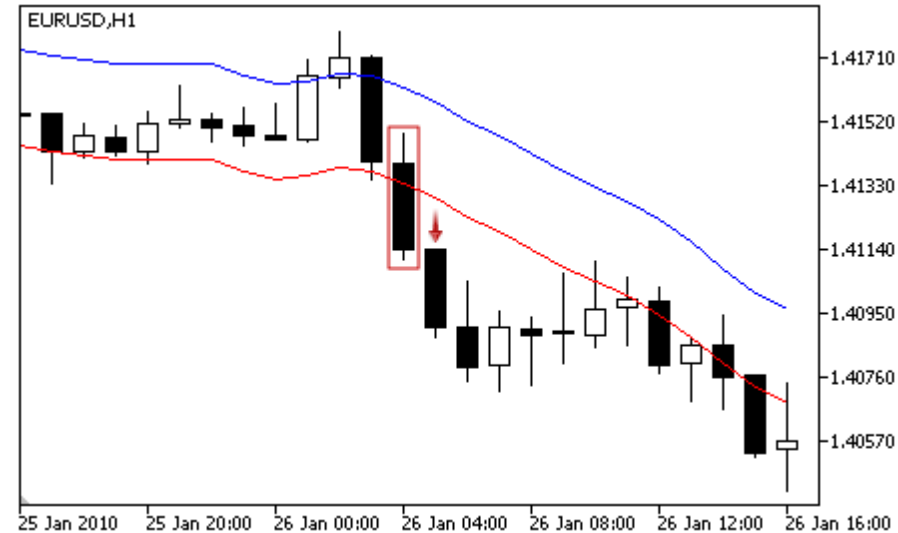
## Signals of the Indicator Envelopes

This module of signals is based on the market models of the indicator [Envelopes](#). The mechanism of making trade decisions based on signals obtained from the modules is described in a [separate section](#).

### Conditions of Generation of Signals

Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

| Signal Type | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                  |
|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| For buying  | <ul style="list-style-type: none"> <li>The price is near the lower line of the indicator at the analyzed bar.</li> </ul>  <ul style="list-style-type: none"> <li>The price crossed the upper line of the indicator at the analyzed bar.</li> </ul>  |
| For selling | <ul style="list-style-type: none"> <li>The price is near the upper line of the indicator at the analyzed bar.</li> </ul>                                                                                                                                                                                                                                                                                                   |

| Signal Type              | Description of Conditions                                                                                                                                                                                                                                                                |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                          |  <p>EURUSD,H1</p> <p>• The price crossed the lower line of the indicator at the analyzed bar.</p>  <p>EURUSD,H1</p> |
| No objections to buying  | No signals.                                                                                                                                                                                                                                                                              |
| No objections to selling | No signals.                                                                                                                                                                                                                                                                              |

### Note

Depending on the mode of operation of an Expert Advisor ("Every tick" or "Open prices only") an analyzed bar is either the current bar (with index 0), or the last formed bar (with index 1).

## Adjustable Parameters

This module has the following adjustable parameters:

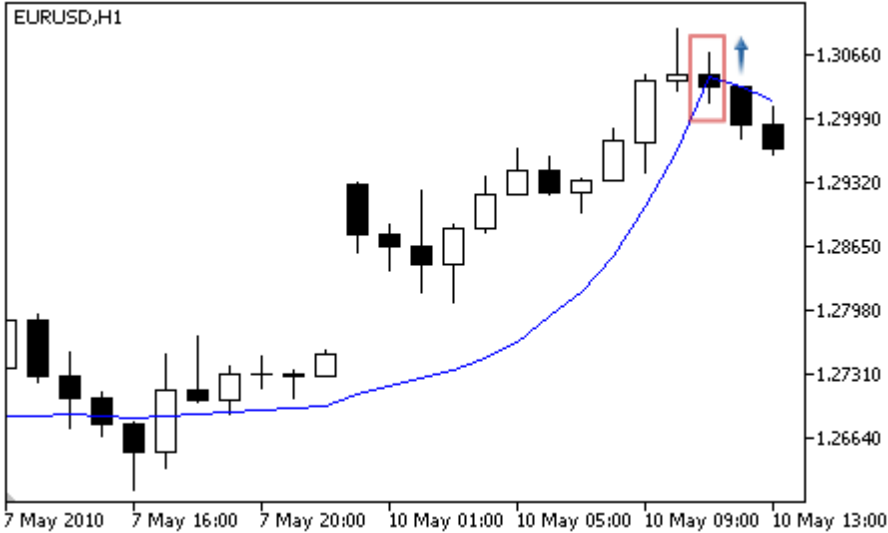
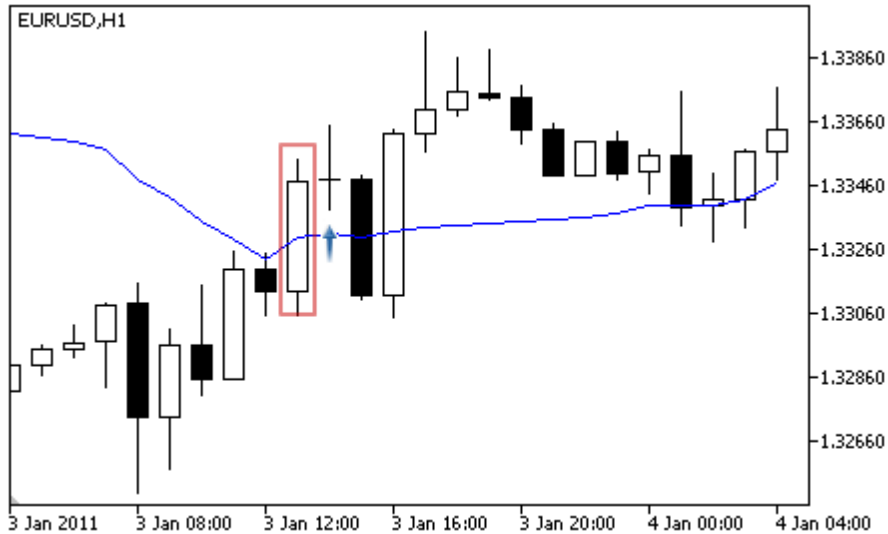
| Parameter | Description                                                                      |
|-----------|----------------------------------------------------------------------------------|
| Weight    | Weight of signal of the module in the interval 0 to 1.                           |
| PeriodMA  | Period of calculation of the indicator.                                          |
| Shift     | Shift of the indicator along the time axis (in bars).                            |
| Method    | <a href="#">Method of averaging</a> .                                            |
| Applied   | A <a href="#">price series</a> used for calculation of the indicator.            |
| Deviation | Deviation of the envelope borders from the center line (MA) in percentage terms. |

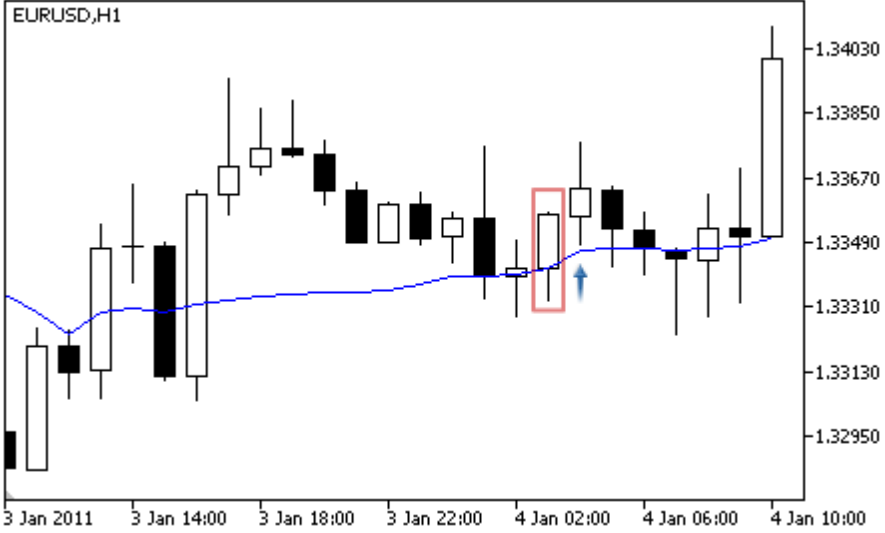
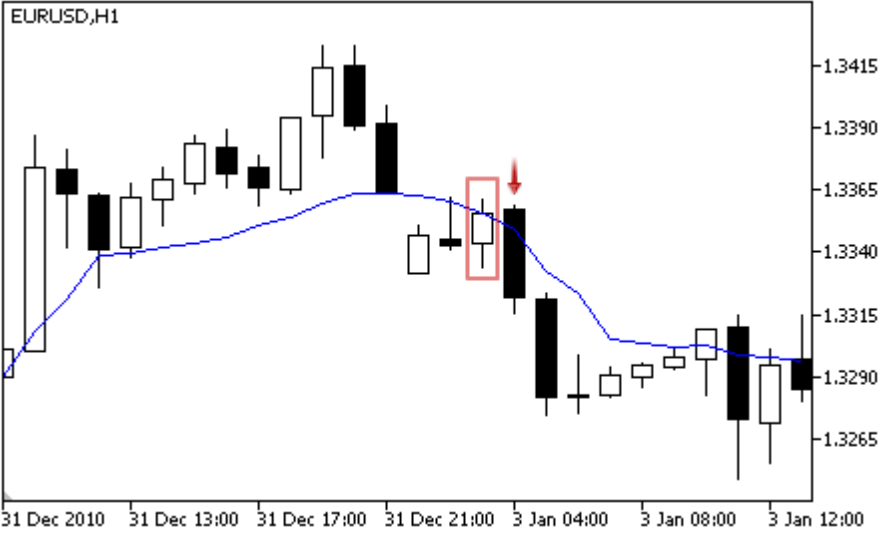
## Signals of the Indicator Fractal Adaptive Moving Average

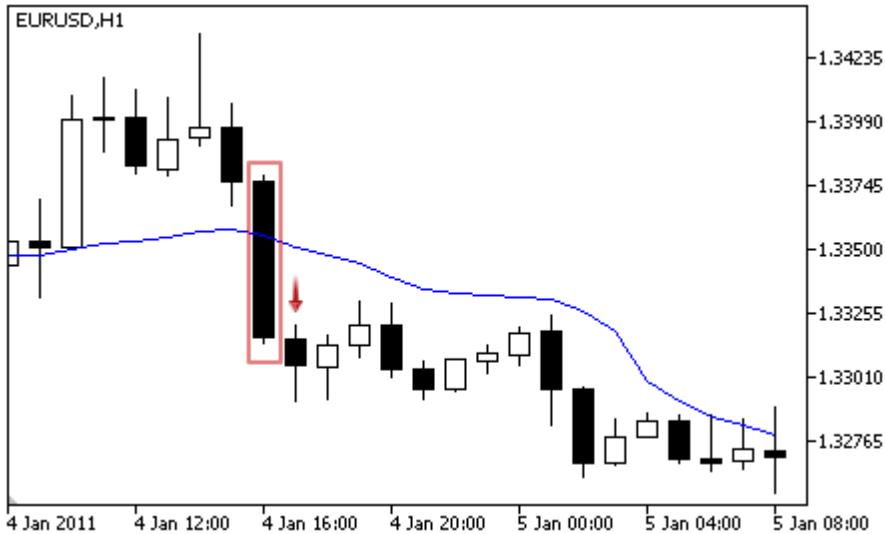
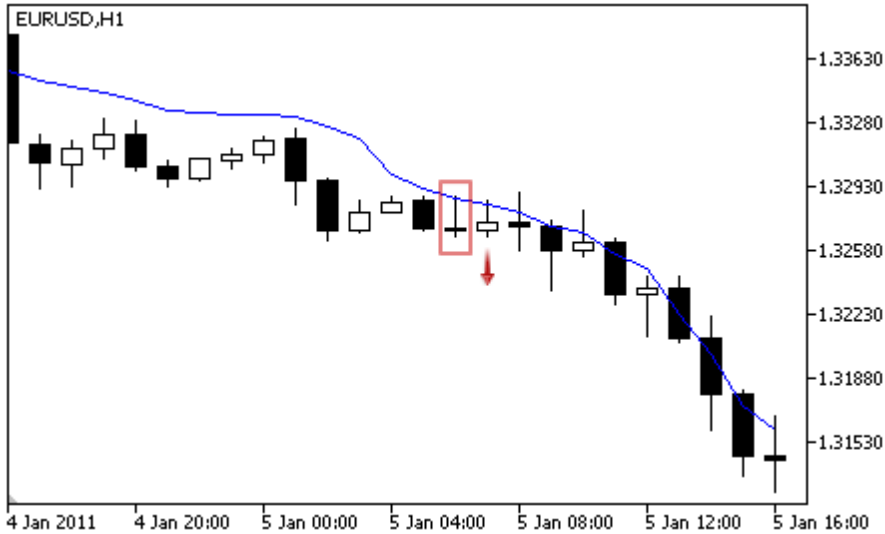
This module of signals is based on the market models of the indicator [Fractal Adaptive Moving Average](#). The mechanism of making trade decisions based on signals obtained from the modules is described in a [separate section](#).

### Conditions of Generation of Signals

Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

| Signal Type | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| For buying  | <ul style="list-style-type: none"> <li>The price has crossed the indicator downwards (the Open price of the analyzed bar is above the indicator and the Close price is below the indicator) and the indicator rises (weak signal).</li> </ul>  <ul style="list-style-type: none"> <li><b>Moving Average crossover.</b> The price has crossed the indicator upwards (the Open price of the analyzed bar is below the indicator and the Close price is above the indicator) and the indicator rises (strong signal).</li> </ul>  |

| Signal Type | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|             | <ul style="list-style-type: none"> <li>The lower shadow of the bar has crossed the indicator (the Open and Close prices of the analyzed bar is above the indicator, and the Low price is below the indicator) and the indicator rises (weak signal).</li> </ul>                                                                                                                                                                                                                                                                 |
| For selling | <ul style="list-style-type: none"> <li>The price has crossed the indicator upwards (the Open price of the analyzed bar is below the indicator and the Close price is above the indicator) and the indicator falls (weak signal).</li> </ul>  <ul style="list-style-type: none"> <li><b>Moving Average crossover.</b> The price has crossed the indicator downwards (the Open price of the analyzed bar is above the indicator and the Close price is below the indicator) and the indicator falls (strong signal).</li> </ul> |

| Signal Type              | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                               |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                          |  <ul style="list-style-type: none"> <li>The upper shadow of the bar has crossed the indicator (the Open and Close prices of the analyzed bar is below the indicator, and the High price is above the indicator) and the indicator falls (weak signal).</li> </ul>  |
| No objections to buying  | The prices is above the indicator.                                                                                                                                                                                                                                                                                                                                                                                                      |
| No objections to selling | The price is below the indicator.                                                                                                                                                                                                                                                                                                                                                                                                       |

#### Note

Depending on the mode of operation of an Expert Advisor ("Every tick" or "Open prices only") an analyzed bar is either the current bar (with index 0), or the last formed bar (with index 1).

## Adjustable Parameters

This module has the following adjustable parameters:

| Parameter | Description                                                           |
|-----------|-----------------------------------------------------------------------|
| Weight    | Weight of signal of the module in the interval 0 to 1.                |
| PeriodMA  | Period of averaging of the indicator.                                 |
| Shift     | Shift of the indicator along the time axis (in bars).                 |
| Method    | <a href="#">Method of averaging</a> .                                 |
| Applied   | A <a href="#">price series</a> used for calculation of the indicator. |



## Signals of the Intraday Time Filter

This module is based on the assumption that the efficiency of market models changes in time. Using this module, you can filter signals received from the other modules by hour and days of week. It allows increasing the quality of generated signals due to cutting off the unfavorable time periods. The mechanism of making trade decisions on the basis of signals of the modules is described in a [separate section](#).

## Conditions of Generation of Signals

Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

| Signal Type              | Description of Conditions                                |
|--------------------------|----------------------------------------------------------|
| For buying               | No signals.                                              |
| For selling              | No signals.                                              |
| No objections to buying  | The current date and time meet the specified parameters. |
| No objections to selling | The current date and time meet the specified parameters. |

## Adjustable Parameters

This module has the following adjustable parameters:

| Parameter     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Weight        | Weight of signal of the module in the interval 0 to 1.                                                                                                                                                                                                                                                                                                                                                                                                           |
| GoodHourOfDay | Number of the only hour of day (from 0 to 23) when trade signals will be enabled. If the value is -1, the signals will be enabled through the whole day.                                                                                                                                                                                                                                                                                                         |
| BadHoursOfDay | The bit field. Each bit of this field corresponds to an hour of day (0 bit - 0 hour, ..., 23 bit - 23-rd hour). If the value of a bit is equal to 0, trade signals will be enabled during the corresponding hour. If the value of a bit is equal to 1, trade signals will be disabled during the corresponding hour. A specified number is represented as a binary number and is used as bit mask.<br>Disabled hours have higher priority than the enabled ones. |
| GoodDayOfWeek | Number of the only day of week (from 0 to 6, where 0 is Sunday), when trade signals will be                                                                                                                                                                                                                                                                                                                                                                      |

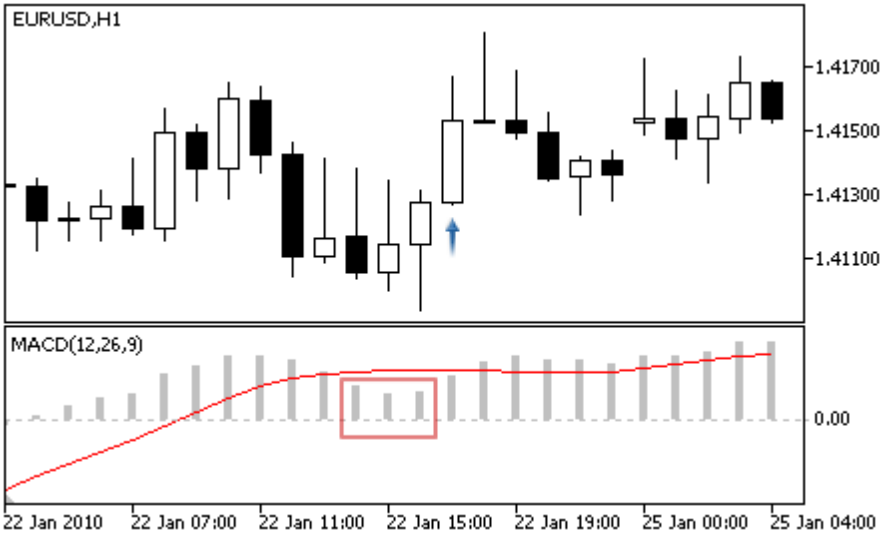
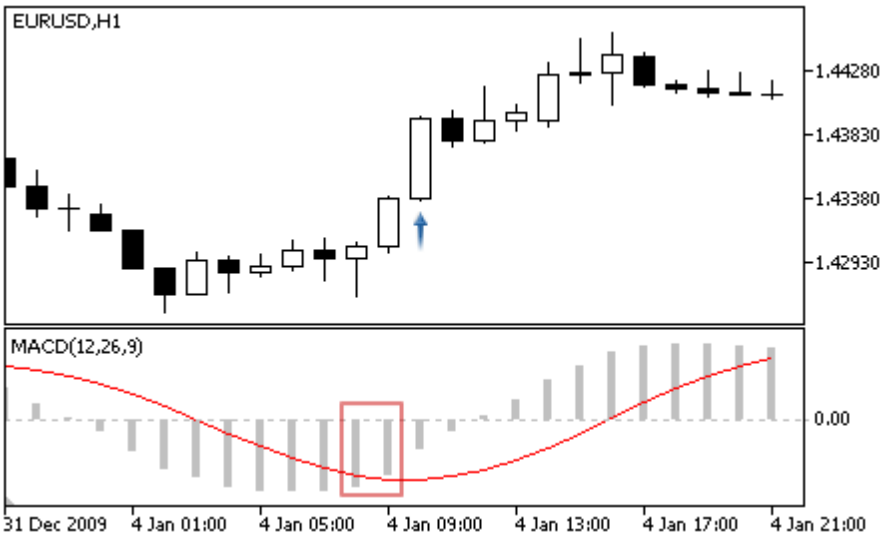
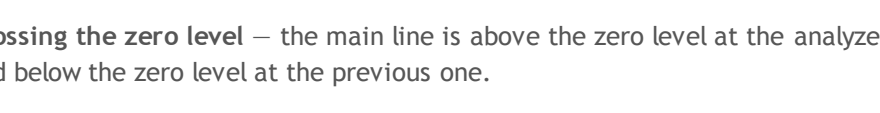
| Parameter     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|               | enabled. If the value is -1, the signals will be enabled on any day.                                                                                                                                                                                                                                                                                                                                                                                                 |
| BadDaysOfWeek | <p>The bit field. Each bit of this field corresponds to a day of week (0 bit - Sunday, ..., 6 bit - Saturday). If the value of a bit is equal to 0, trade signals will be enabled during the corresponding day. If the value of a bit is equal to 1, trade signals will be disabled during the corresponding day. A specified number is represented as a binary number and is used as bit mask.</p> <p>Disabled days have higher priority than the enabled ones.</p> |

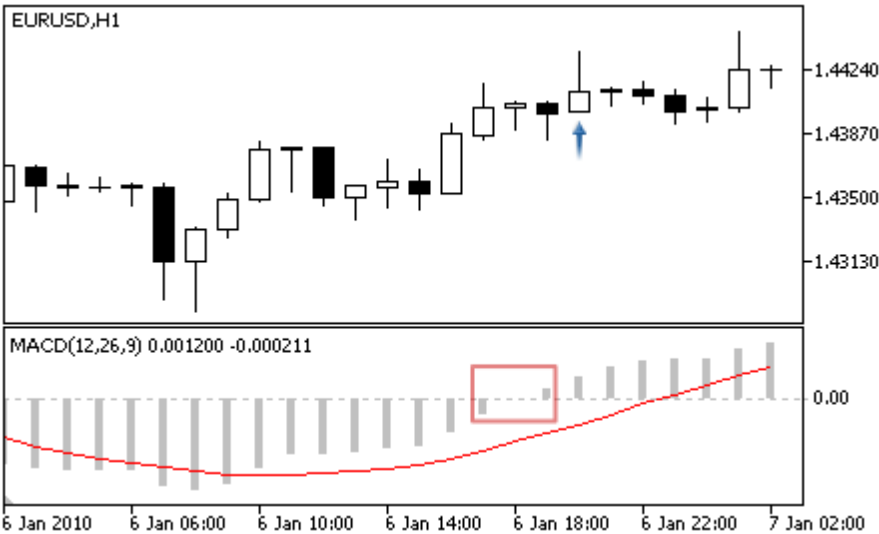
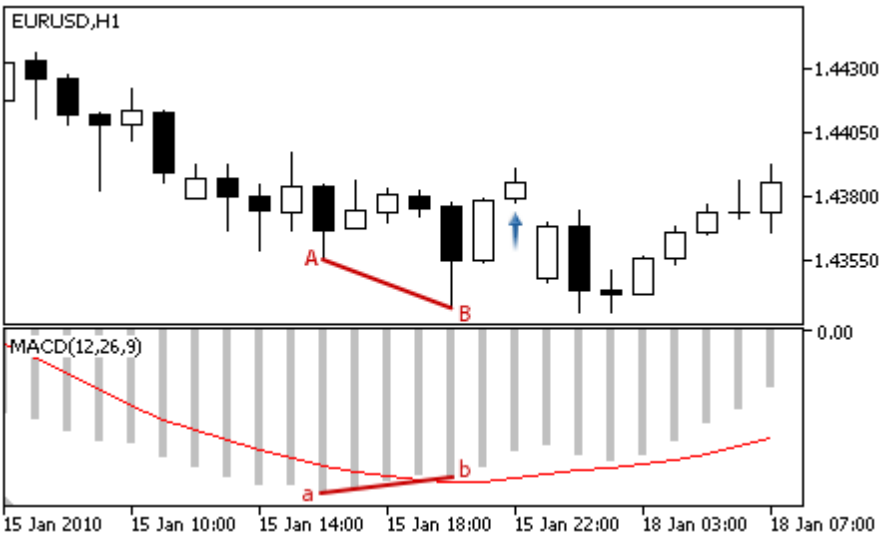
## Signals of the Oscillator MACD

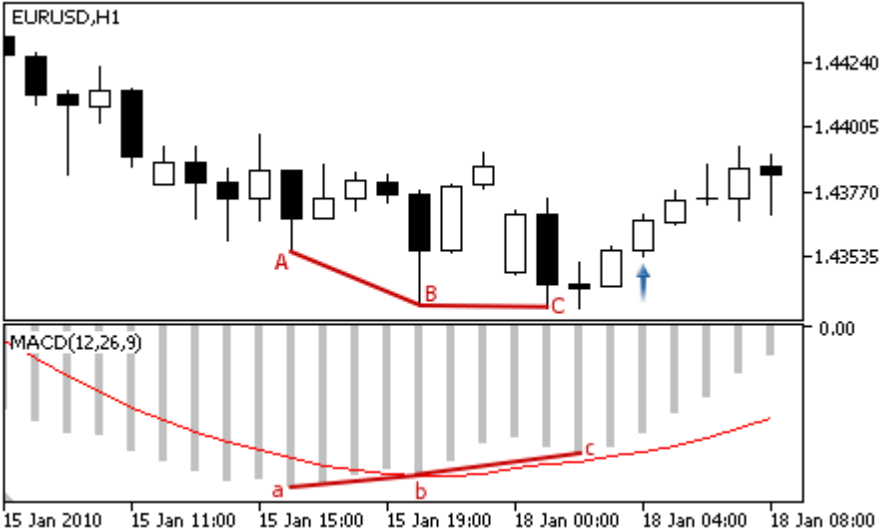
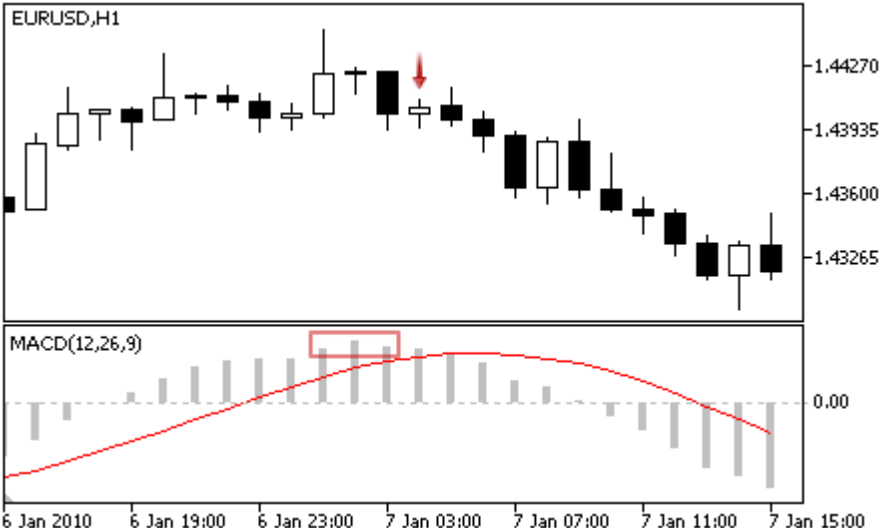
This module of signals is based on the market models of the oscillator [MACD](#). The mechanism of making trade decisions based on signals obtained from the modules is described in a [separate section](#).

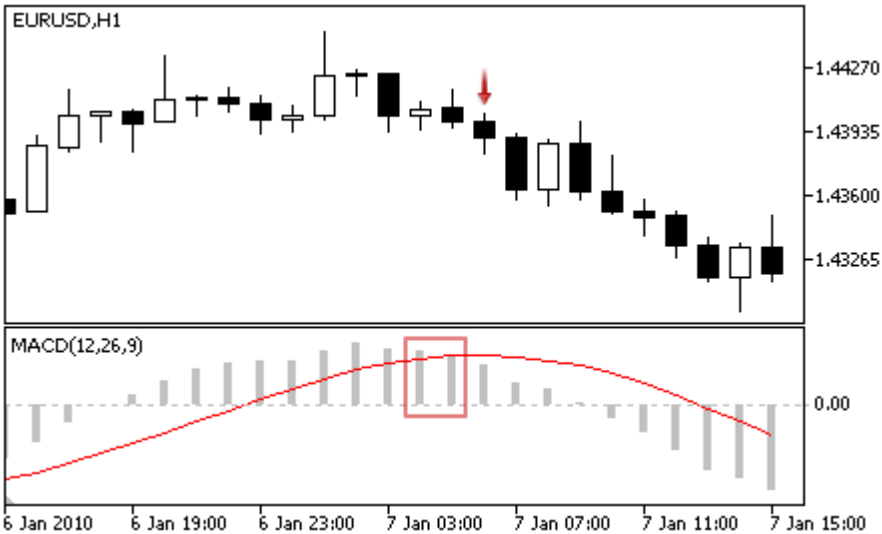
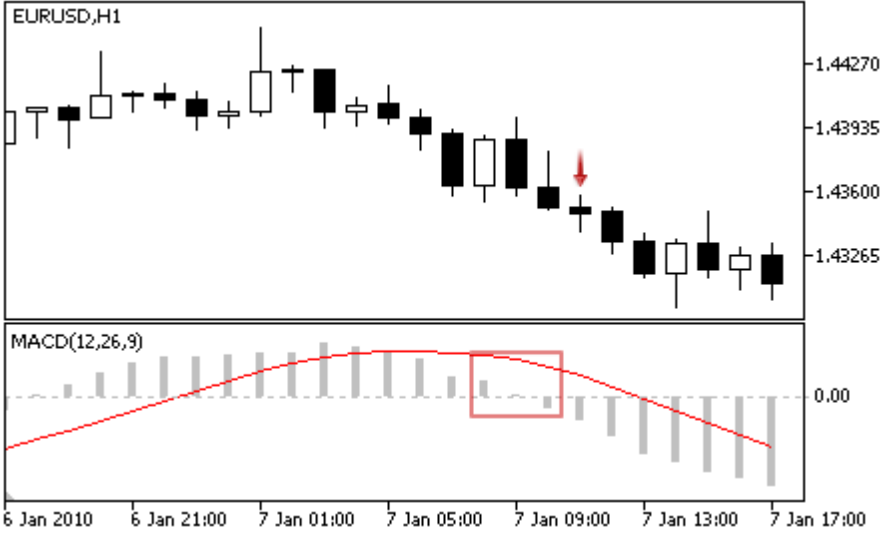
### Conditions of Generation of Signals

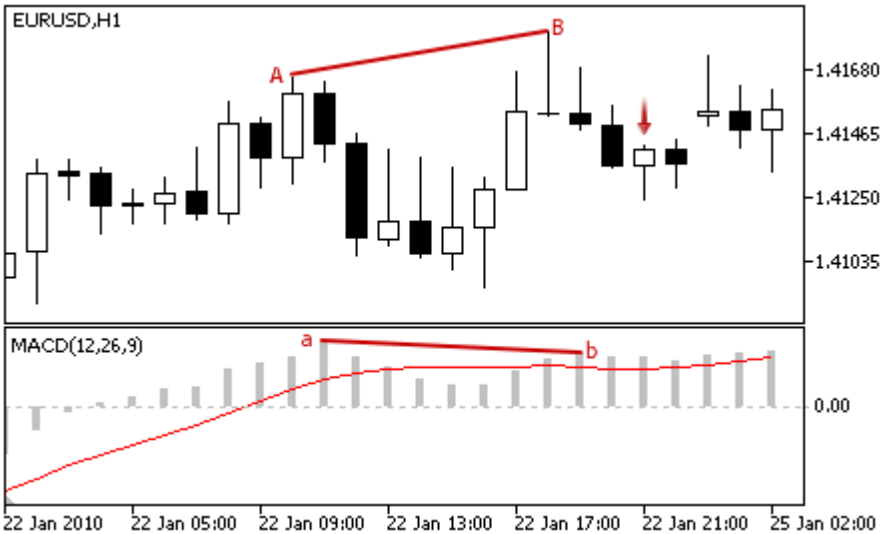
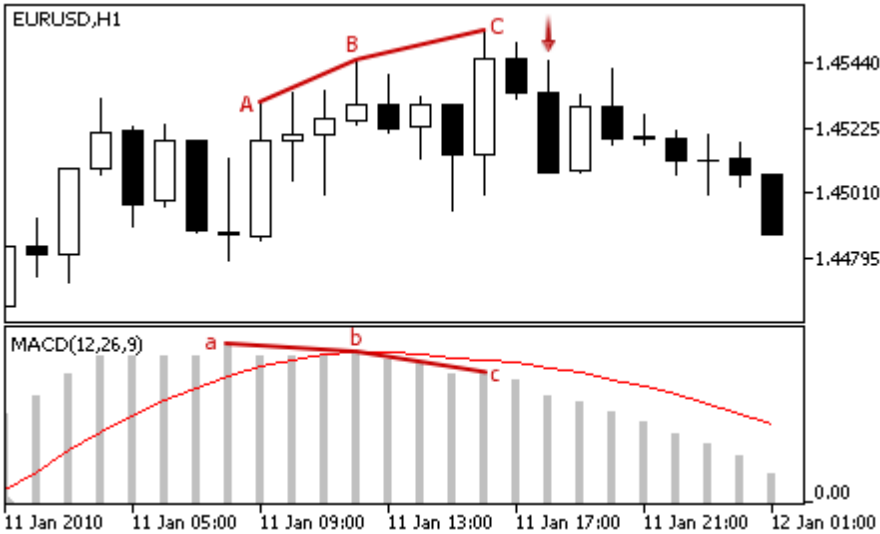
Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

| Signal Type | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| For buying  | <ul style="list-style-type: none"> <li> <b>Reverse</b> – the oscillator turned upwards (the oscillator rises at the analyzed bar and falls at the previous one).            </li> <li> <b>Crossover of the main and signal line</b> – the main line is above the signal line at the analyzed bar and below the signal line at the previous one.            </li> <li> <b>Crossing the zero level</b> – the main line is above the zero level at the analyzed bar and below the zero level at the previous one.            </li> </ul> |

| Signal Type | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|             | <div data-bbox="438 315 1321 846">  <p>EURUSD,H1</p> <p>MACD(12,26,9) 0.001200 -0.000211</p> <p>6 Jan 2010 6 Jan 06:00 6 Jan 10:00 6 Jan 14:00 6 Jan 18:00 6 Jan 22:00 7 Jan 02:00</p> </div> <ul style="list-style-type: none"> <li>• <b>Divergence</b> – the first analyzed bottom of the oscillator is higher than the previous one, and the corresponding price bottom is lower than the previous one.</li> </ul> <div data-bbox="438 958 1321 1489">  <p>EURUSD,H1</p> <p>MACD(12,26,9)</p> <p>15 Jan 2010 15 Jan 10:00 15 Jan 14:00 15 Jan 18:00 15 Jan 22:00 18 Jan 03:00 18 Jan 07:00</p> </div> <ul style="list-style-type: none"> <li>• <b>Double divergence</b> – the oscillator form three consequent bottoms, each of them is higher than the previous one; and the price formed three corresponding bottoms, and each of them is lower than the previous one.</li> </ul> |

| Signal Type | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|             |  <p>EURUSD, H1</p> <p>MACD(12,26,9)</p> <p>15 Jan 2010 15 Jan 11:00 15 Jan 15:00 15 Jan 19:00 18 Jan 00:00 18 Jan 04:00 18 Jan 08:00</p>                                                                                                                                                                                                                                                                                                                                                                                       |
| For selling | <ul style="list-style-type: none"> <li>• <b>Reverse</b> – the oscillator turned downwards (the oscillator falls at the analyzed bar and rises at the previous one).</li> </ul>  <p>EURUSD, H1</p> <p>MACD(12,26,9)</p> <p>6 Jan 2010 6 Jan 19:00 6 Jan 23:00 7 Jan 03:00 7 Jan 07:00 7 Jan 11:00 7 Jan 15:00</p> <ul style="list-style-type: none"> <li>• <b>Crossover of the main and signal line</b> – the main line is below the signal line at the analyzed bar and above the signal line at the previous one.</li> </ul> |

| Signal Type | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|             | <div data-bbox="435 315 1321 846">  <p>EURUSD,H1</p> <p>MACD(12,26,9)</p> <p>6 Jan 2010 6 Jan 19:00 6 Jan 23:00 7 Jan 03:00 7 Jan 07:00 7 Jan 11:00 7 Jan 15:00</p> </div> <ul style="list-style-type: none"> <li>• <b>Crossing the zero level</b> – the main line is below the zero level at the analyzed bar and above the zero level at the previous one.</li> </ul> <div data-bbox="435 965 1321 1496">  <p>EURUSD,H1</p> <p>MACD(12,26,9)</p> <p>6 Jan 2010 6 Jan 21:00 7 Jan 01:00 7 Jan 05:00 7 Jan 09:00 7 Jan 13:00 7 Jan 17:00</p> </div> <ul style="list-style-type: none"> <li>• <b>Divergence</b> – the first analyzed peak of the oscillator is lower than the previous one, and the corresponding price peak is higher than the previous peak.</li> </ul> |

| Signal Type              | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                           |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                          |  <p>• <b>Double divergence</b> – the oscillator formed three consequent peaks, each of them is lower than the previous one; and the price formed three corresponding peaks, each of them is higher than the previous one.</p>  |
| No objections to buying  | Value of the oscillator grows at the analyzed bar.                                                                                                                                                                                                                                                                                                                                                  |
| No objections to selling | Value of the oscillator falls at the analyzed bar.                                                                                                                                                                                                                                                                                                                                                  |

#### Note

Depending on the mode of operation of an Expert Advisor ("Every tick" or "Open prices only") an analyzed bar is either the current bar (with index 0), or the last formed bar (with index 1).

## Adjustable Parameters

This module has the following adjustable parameters:

| Parameter    | Description                                                            |
|--------------|------------------------------------------------------------------------|
| Weight       | Weight of signal of the module in the interval 0 to 1.                 |
| PeriodFast   | Period of calculation of the fast EMA.                                 |
| PeriodSlow   | Period of calculation of the slow EMA.                                 |
| PeriodSignal | Period of smoothing.                                                   |
| Applied      | A <a href="#">price series</a> used for calculation of the oscillator. |

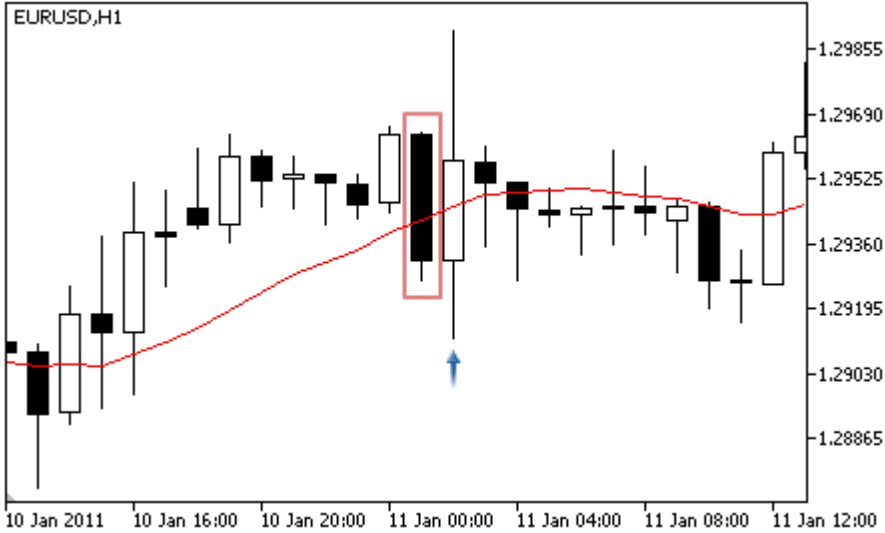
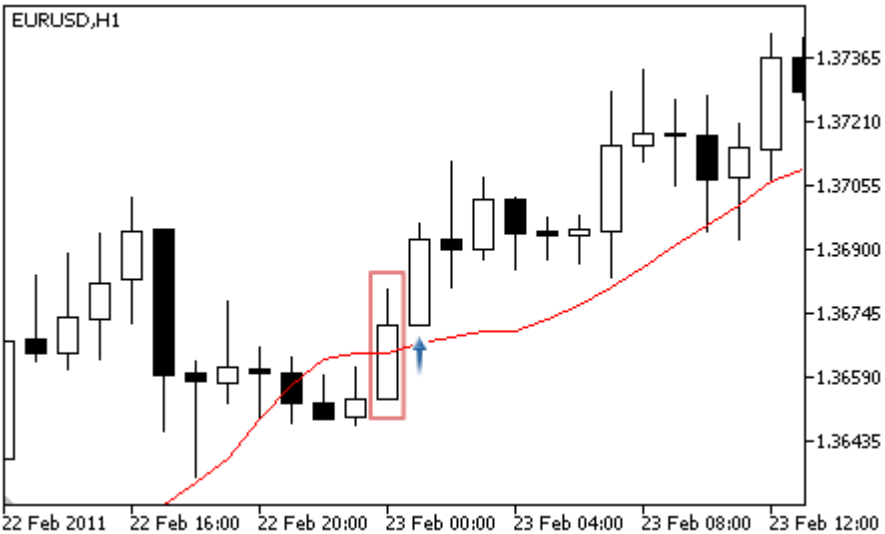


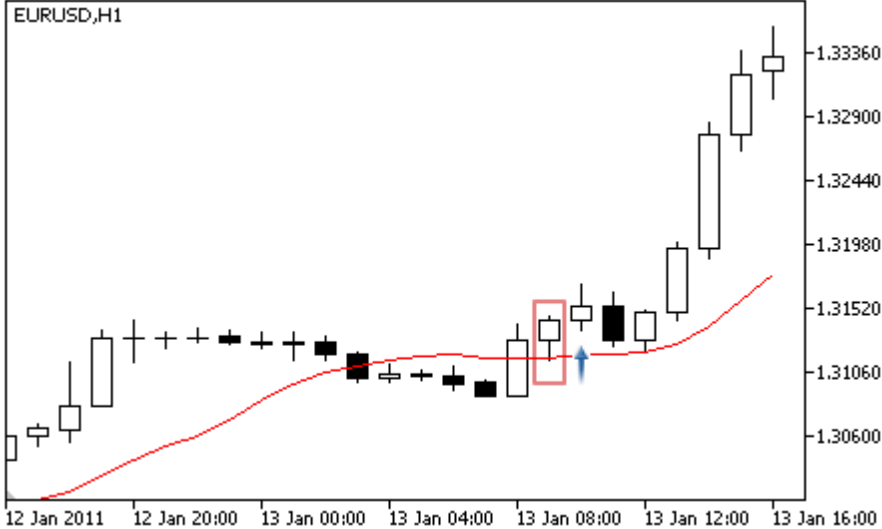
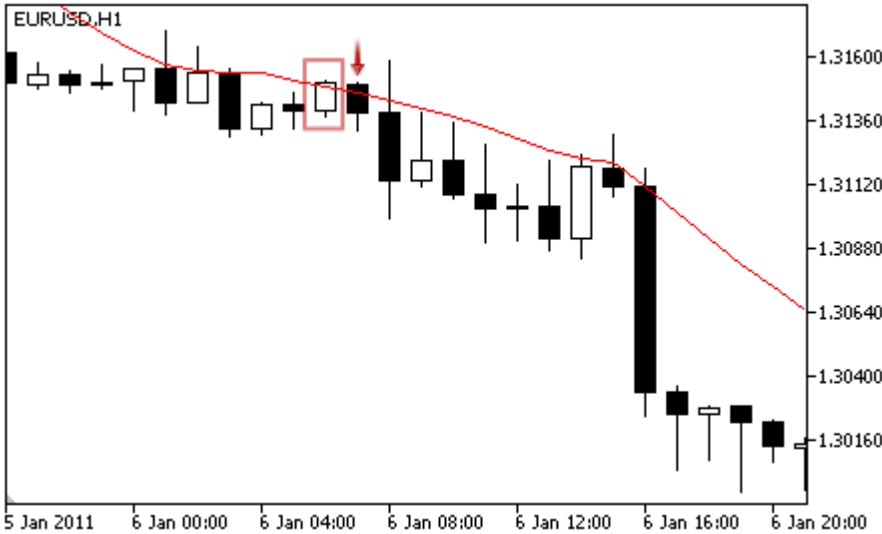
## Signals of the Indicator Moving Average

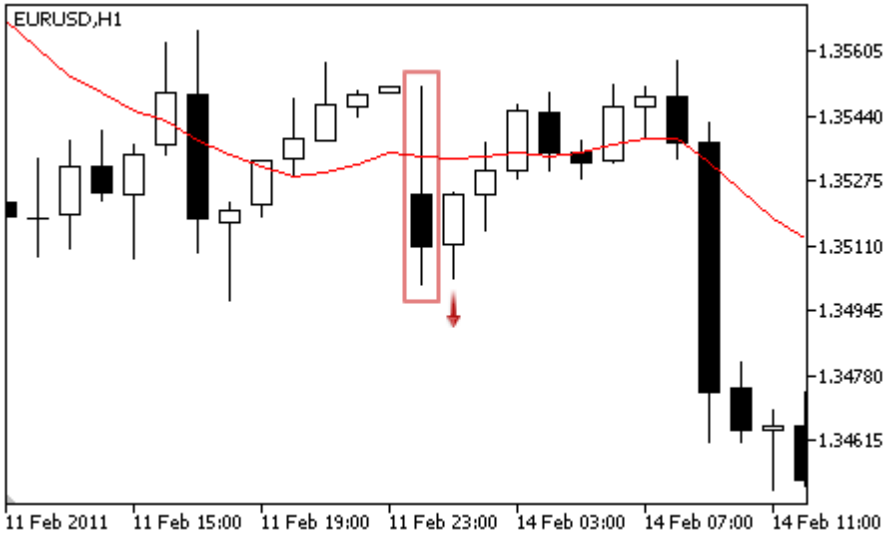
This module of signals is based on the market models of the indicator [Moving Average](#). The mechanism of making trade decisions based on signals obtained from the modules is described in a [separate section](#).

### Conditions of Generation of Signals

Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

| Signal Type | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| For buying  | <ul style="list-style-type: none"> <li>The price has crossed the indicator downwards (the Open price of the analyzed bar is above the indicator and the Close price is below the indicator) and the indicator rises (weak signal).</li> </ul>  <ul style="list-style-type: none"> <li><b>Moving Average crossover.</b> The price has crossed the indicator upwards (the Open price of the analyzed bar is below the indicator and the Close price is above the indicator) and the indicator rises (strong signal).</li> </ul>  |

| Signal Type | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|             | <ul style="list-style-type: none"> <li>The lower shadow of the bar has crossed the indicator (the Open and Close prices of the analyzed bar is above the indicator, and the Low price is below the indicator) and the indicator rises (weak signal).</li> </ul>                                                                                                                                                                                                                                                                 |
| For selling | <ul style="list-style-type: none"> <li>The price has crossed the indicator upwards (the Open price of the analyzed bar is below the indicator and the Close price is above the indicator) and the indicator falls (weak signal).</li> </ul>  <ul style="list-style-type: none"> <li><b>Moving Average crossover.</b> The price has crossed the indicator downwards (the Open price of the analyzed bar is above the indicator and the Close price is below the indicator) and the indicator falls (strong signal).</li> </ul> |

| Signal Type              | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                               |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                          |  <ul style="list-style-type: none"> <li>The upper shadow of the bar has crossed the indicator (the Open and Close prices of the analyzed bar is below the indicator, and the High price is above the indicator) and the indicator falls (weak signal).</li> </ul>  |
| No objections to buying  | The prices is above the indicator.                                                                                                                                                                                                                                                                                                                                                                                                      |
| No objections to selling | The price is below the indicator.                                                                                                                                                                                                                                                                                                                                                                                                       |

#### Note

Depending on the mode of operation of an Expert Advisor ("Every tick" or "Open prices only") an analyzed bar is either the current bar (with index 0), or the last formed bar (with index 1).

## Adjustable Parameters

This module has the following adjustable parameters:

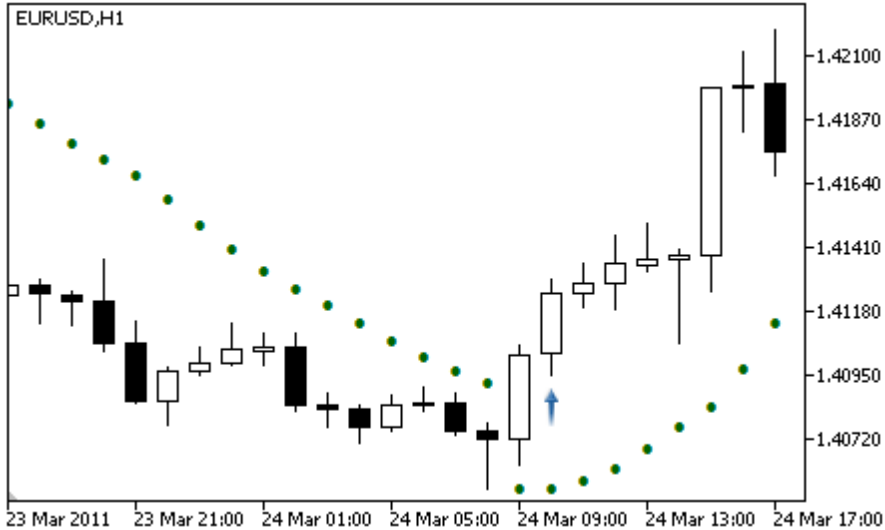
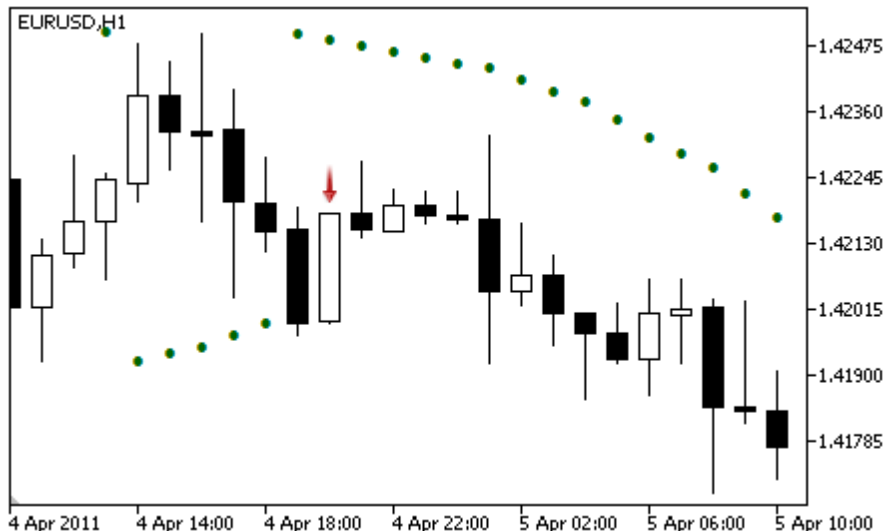
| Parameter | Description                                                           |
|-----------|-----------------------------------------------------------------------|
| Weight    | Weight of signal of the module in the interval 0 to 1.                |
| PeriodMA  | Period of averaging of the indicator.                                 |
| Shift     | Shift of the indicator along the time axis (in bars).                 |
| Method    | <a href="#">Method of averaging</a> .                                 |
| Applied   | A <a href="#">price series</a> used for calculation of the indicator. |

## Signals of the indicator Parabolic SAR

This module of signals is based on the market models of the indicator [Parabolic SAR](#). The mechanism of making trade decisions based on signals obtained from the modules is described in a [separate section](#).

### Conditions of Generation of Signals

Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

| Signal Type   | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| For buying    | <p><b>Reverse</b> — the indicator is below the price at the analyzed bar and above the price at the previous one.</p>  <p>The chart displays EURUSD on an H1 timeframe from March 23 to 24, 2011. The price is shown as a candlestick chart, and the Parabolic SAR indicator is represented by green dots. A blue arrow points to a 'Reverse' buy signal where the price is below the indicator and above the previous bar's price.</p> |
| For selling   | <p><b>Reverse</b> — the indicator is above the price at the analyzed bar and below the price at the previous one.</p>  <p>The chart displays EURUSD on an H1 timeframe from April 4 to 5, 2011. The price is shown as a candlestick chart, and the Parabolic SAR indicator is represented by green dots. A red arrow points to a 'Reverse' sell signal where the price is above the indicator and below the previous bar's price.</p>  |
| No objections | The prices is above the indicator.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

| Signal Type              | Description of Conditions         |
|--------------------------|-----------------------------------|
| to buying                |                                   |
| No objections to selling | The price is below the indicator. |

**Note**

Depending on the mode of operation of an Expert Advisor ("Every tick" or "Open prices only") an analyzed bar is either the current bar (with index 0), or the last formed bar (with index 1).

## Adjustable Parameters

This module has the following adjustable parameters:

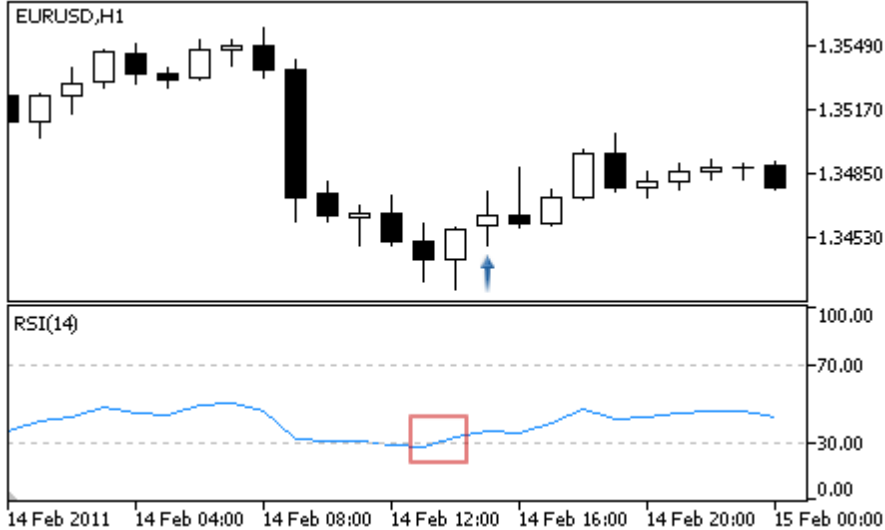
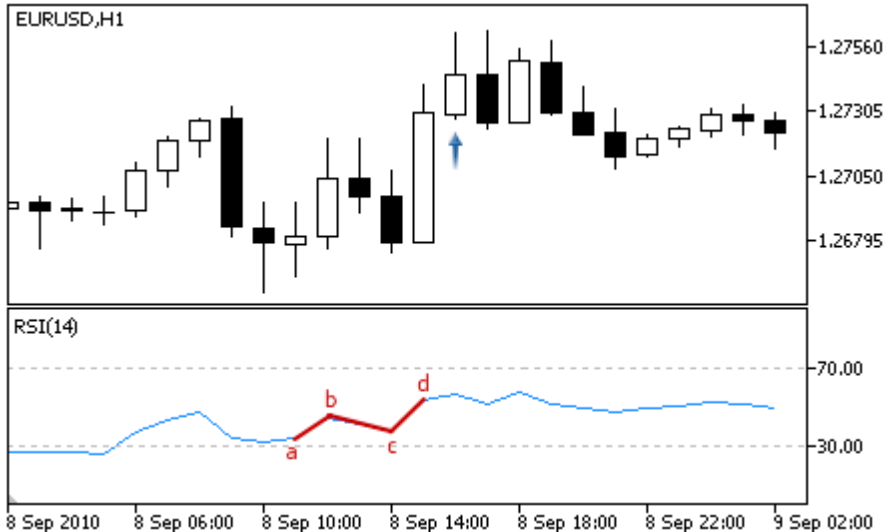
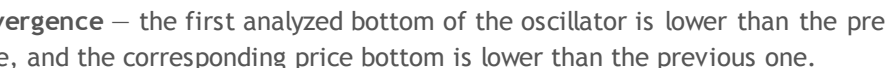
| Parameter | Description                                                               |
|-----------|---------------------------------------------------------------------------|
| Weight    | Weight of signal of the module in the interval 0 to 1.                    |
| Step      | The increment of speed of the indicator.                                  |
| Maximum   | Maximum rate of the speed of convergence of the indicator with the price. |

## Signals of the Oscillator Relative Strength Index

This module of signals is based on the market models of the oscillator [Relative Strength Index](#). The mechanism of making trade decisions based on signals obtained from the modules is described in a [separate section](#).

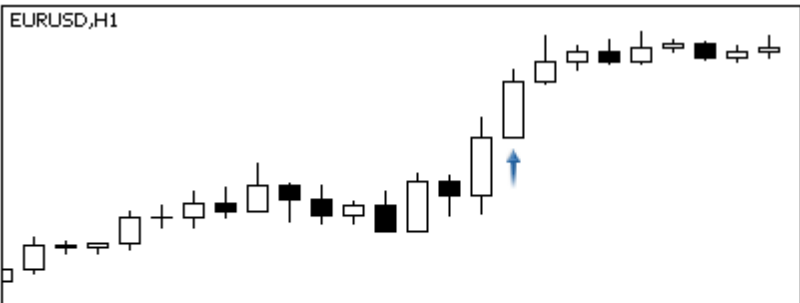
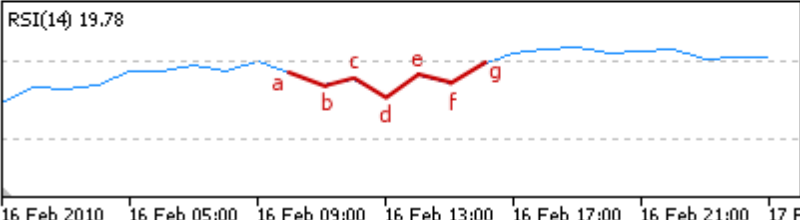
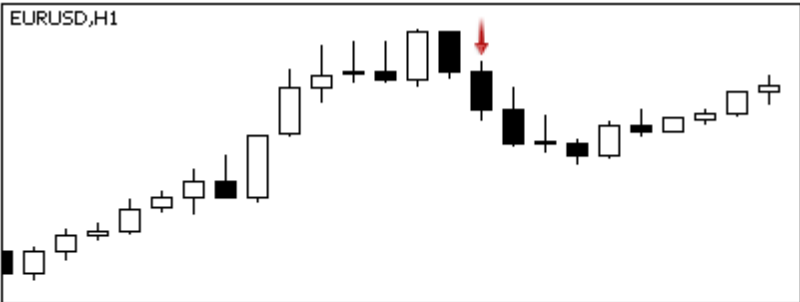
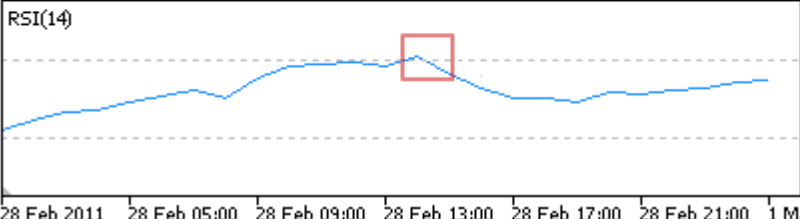
### Conditions of Generation of Signals

Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

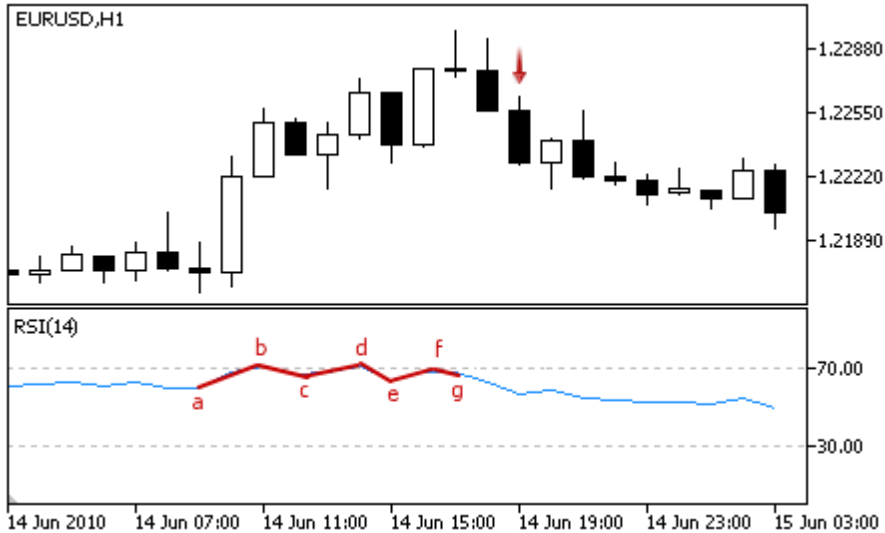
| Signal Type | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| For buying  | <ul style="list-style-type: none"> <li> <b>Reverse behind the oversold level</b> – the oscillator turned upwards and its value at the analyzed bar is behind the oversold level (default value is 30).            </li> <li> <b>Failed swing</b> – the oscillator rises higher than the previous peak at the analyzed bar.            </li> <li> <b>Divergence</b> – the first analyzed bottom of the oscillator is lower than the previous one, and the corresponding price bottom is lower than the previous one.            </li> </ul> |

| Signal Type | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|             | <div data-bbox="432 315 1321 846"> <p>EURUSD, H1</p> <p>RSI(14)</p> <p>20 Aug 2010 20 Aug 10:00 20 Aug 14:00 20 Aug 18:00 20 Aug 22:00 23 Aug 02:00 23 Aug 06:00</p> </div> <ul style="list-style-type: none"> <li>• <b>Double divergence</b> – the oscillator form three consequent bottoms, each of them is higher than the previous one; and the price formed three corresponding bottoms, and each of them is lower than the previous one.</li> </ul> <div data-bbox="432 994 1321 1525"> <p>EURUSD, H1</p> <p>RSI(14)</p> <p>11 Nov 2010 11 Nov 19:00 11 Nov 23:00 12 Nov 03:00 12 Nov 07:00 12 Nov 11:00 12 Nov 15:00</p> </div> <ul style="list-style-type: none"> <li>• <b>Head/Shoulders</b> – the oscillator formed three consequent bottoms, and the mid one is lower than the others.</li> </ul> |



| Signal Type | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|             |   <p>EURUSD,H1</p> <p>RSI(14) 19.78</p> <p>16 Feb 2010 16 Feb 05:00 16 Feb 09:00 16 Feb 13:00 16 Feb 17:00 16 Feb 21:00 17 Feb 01:00</p>                                                                                                                                                                                                                                                                                                                                                                         |
| For selling | <ul style="list-style-type: none"> <li>• <b>Reverse behind the overbought level</b> – the oscillator turned downwards and its value at the analyzed bar is behind the overbought level (default value is 70).</li> </ul>   <p>EURUSD,H1</p> <p>RSI(14)</p> <p>28 Feb 2011 28 Feb 05:00 28 Feb 09:00 28 Feb 13:00 28 Feb 17:00 28 Feb 21:00 1 Mar 01:00</p> <ul style="list-style-type: none"> <li>• <b>Failed swing</b> – the oscillator falls lower than the previous bottom at the analyzed bar.</li> </ul> |

| Signal Type | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|             | <div data-bbox="432 315 1318 846"> <p>EURUSD,H1</p> <p>RSI(14)</p> <p>23 Aug 2010 23 Aug 06:00 23 Aug 10:00 23 Aug 14:00 23 Aug 18:00 23 Aug 22:00 24 Aug 02:00</p> </div> <ul style="list-style-type: none"> <li>• <b>Divergence</b> – the first analyzed peak of the oscillator is lower than the previous one, and the corresponding price peak is higher than the previous peak.</li> </ul> <div data-bbox="432 969 1318 1500"> <p>EURUSD,H1</p> <p>RSI(14)</p> <p>21 Sep 2010 22 Sep 02:00 22 Sep 06:00 22 Sep 10:00 22 Sep 14:00 22 Sep 18:00 22 Sep 22:00</p> </div> <ul style="list-style-type: none"> <li>• <b>Double divergence</b> – the oscillator formed three consequent peaks, each of them is lower than the previous one; and the price formed three corresponding peaks, each of them is higher than the previous one.</li> </ul> |

| Signal Type              | Description of Conditions                                                                                                                                                                                                                                                                        |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                          |  <p>• <b>Head/Shoulders</b> – the oscillator formed three consequent peaks, and the mid one is higher than the others.</p>  |
| No objections to buying  | Value of the oscillator grows at the analyzed bar.                                                                                                                                                                                                                                               |
| No objections to selling | Value of the oscillator falls at the analyzed bar.                                                                                                                                                                                                                                               |

#### Note

Depending on the mode of operation of an Expert Advisor ("Every tick" or "Open prices only") an analyzed bar is either the current bar (with index 0), or the last formed bar (with index 1).

## Adjustable Parameters

This module has the following adjustable parameters:

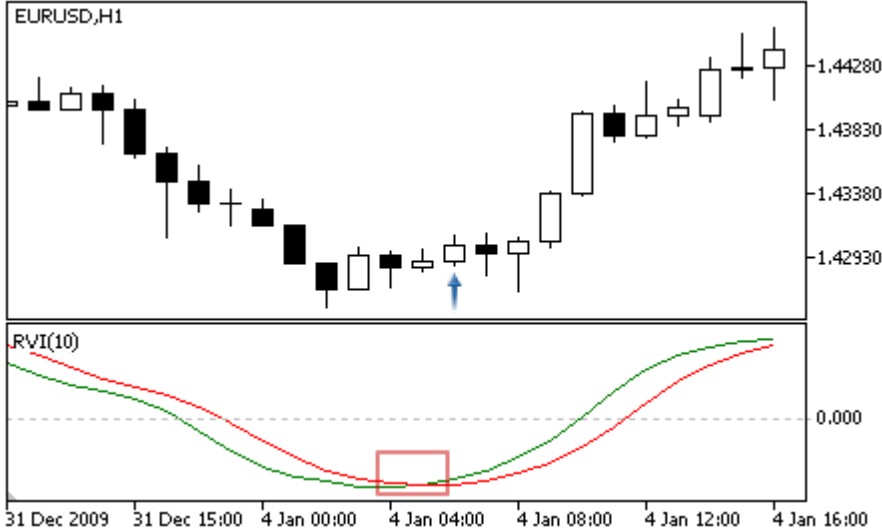
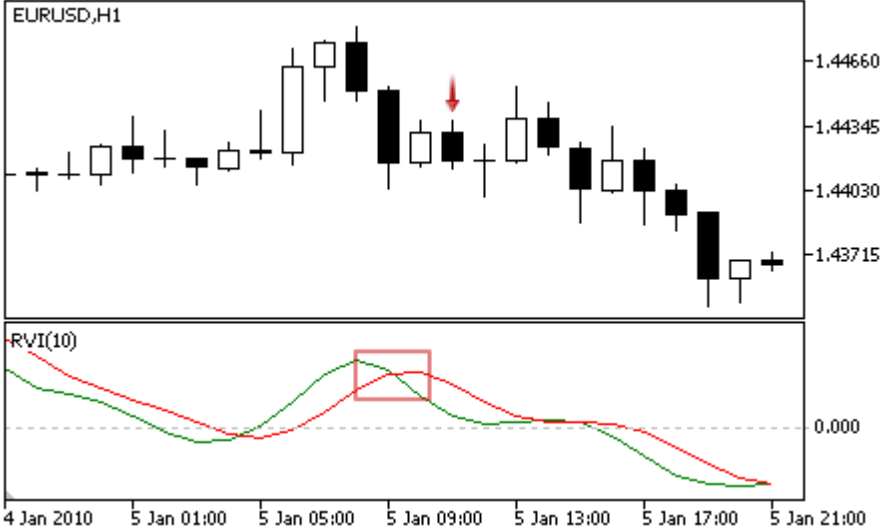
| Parameter | Description                                                            |
|-----------|------------------------------------------------------------------------|
| Weight    | Weight of signal of the module in the interval 0 to 1.                 |
| PeriodRSI | Period of calculation of the oscillator.                               |
| Applied   | A <a href="#">price series</a> used for calculation of the oscillator. |

## Signals of the Oscillator Relative Vigor Index

This module of signals is based on the market models of the oscillator [Relative Vigor Index](#). The mechanism of making trade decisions based on signals obtained from the modules is described in a [separate section](#).

### Conditions of Generation of Signals

Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

| Signal Type   | Description of Conditions                                                                                                                                                                                                                           |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| For buying    | <p><b>Crossing of the main and signal line</b> – the main line is above the signal line at the analyzed bar and below the signal line at the previous one.</p>   |
| For selling   | <p><b>Crossing of the main and signal line</b> – the main line is below the signal line at the analyzed bar and above the signal line at the previous one.</p>  |
| No objections | Value of the oscillator grows at the analyzed bar.                                                                                                                                                                                                  |

| Signal Type              | Description of Conditions                          |
|--------------------------|----------------------------------------------------|
| to buying                |                                                    |
| No objections to selling | Value of the oscillator falls at the analyzed bar. |

**Note**

Depending on the mode of operation of an Expert Advisor ("Every tick" or "Open prices only") an analyzed bar is either the current bar (with index 0), or the last formed bar (with index 1).

## Adjustable Parameters

This module has the following adjustable parameters:

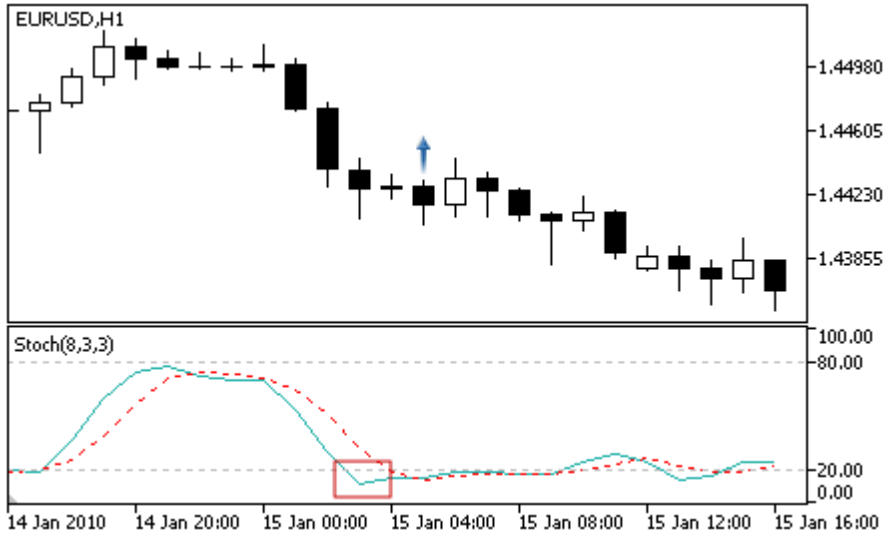
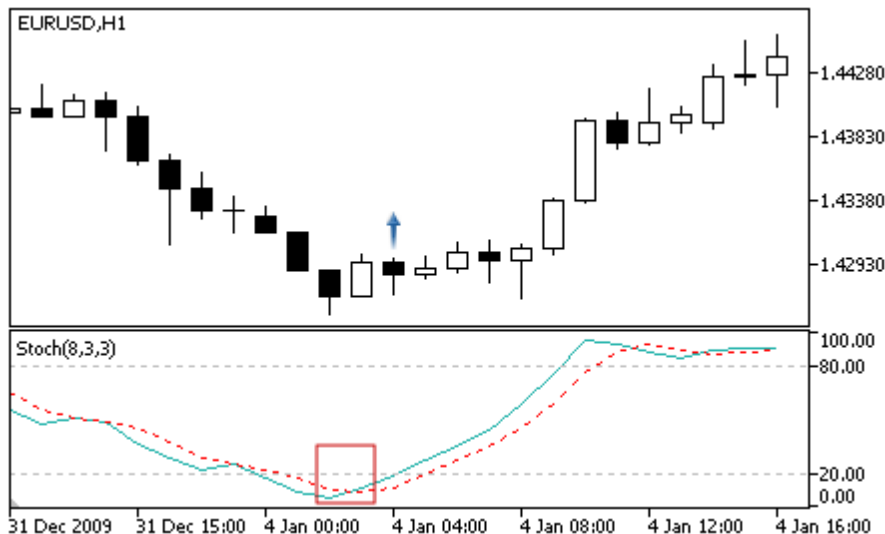
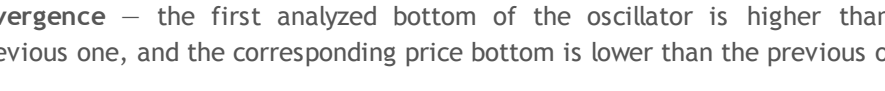
| Parameter | Description                                            |
|-----------|--------------------------------------------------------|
| Weight    | Weight of signal of the module in the interval 0 to 1. |
| PeriodRVI | Period of calculation of the oscillator.               |

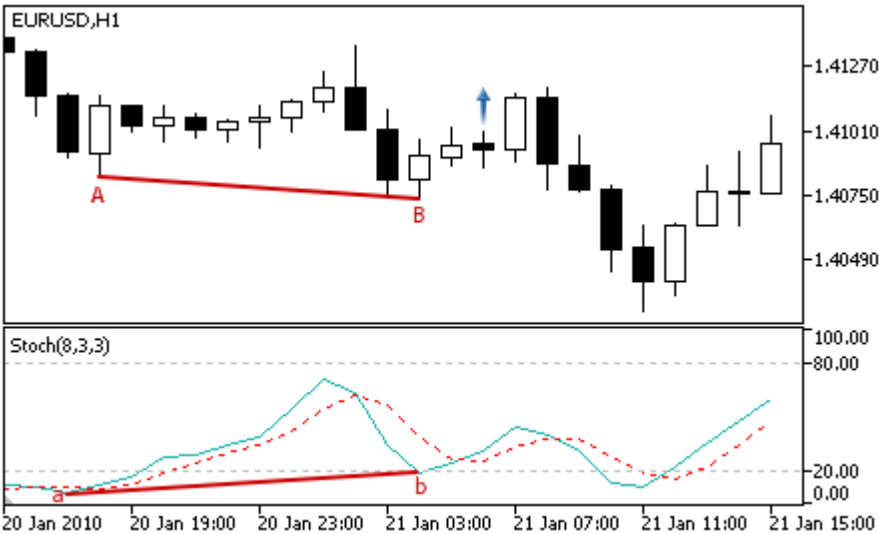
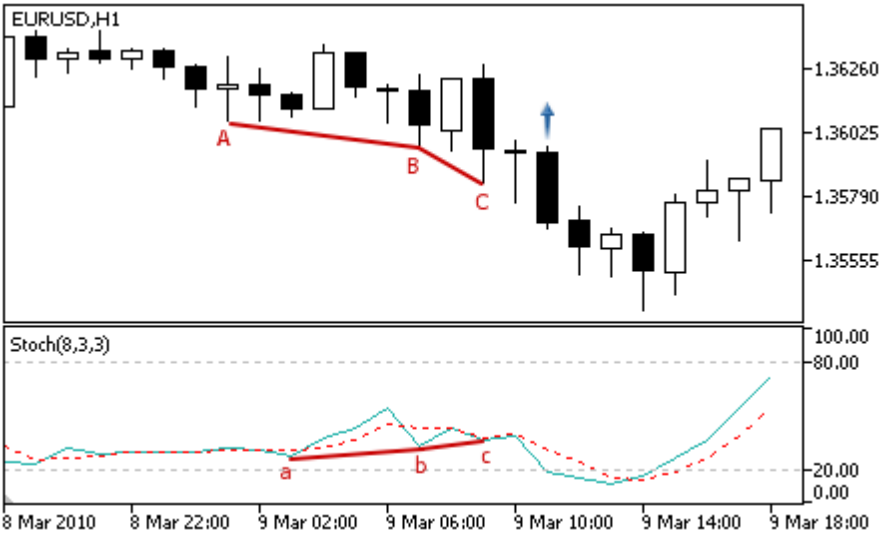
## Signals of the Oscillator Stochastic

This module of signals based on the market models of the oscillator [Stochastic](#). The mechanism of making trade decisions based on signals obtained from the modules is described in a [separate section](#).

### Conditions of Generation of Signals

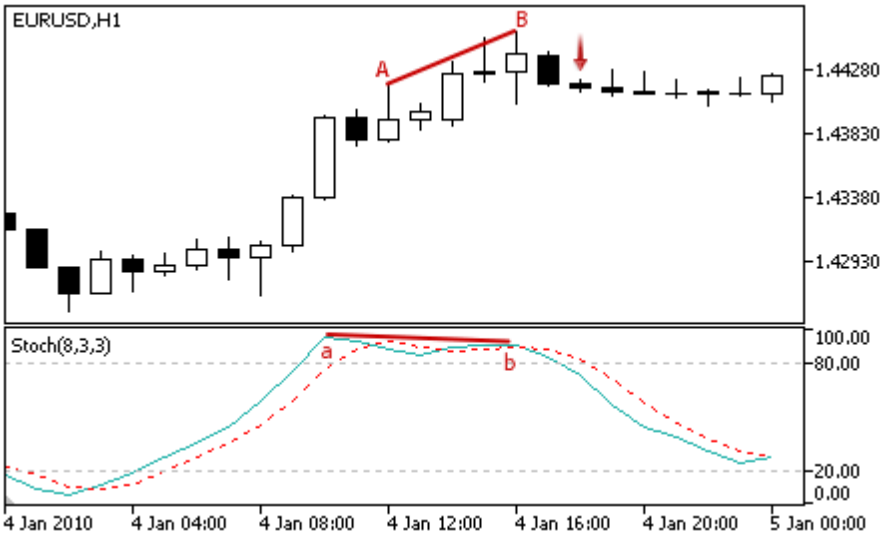
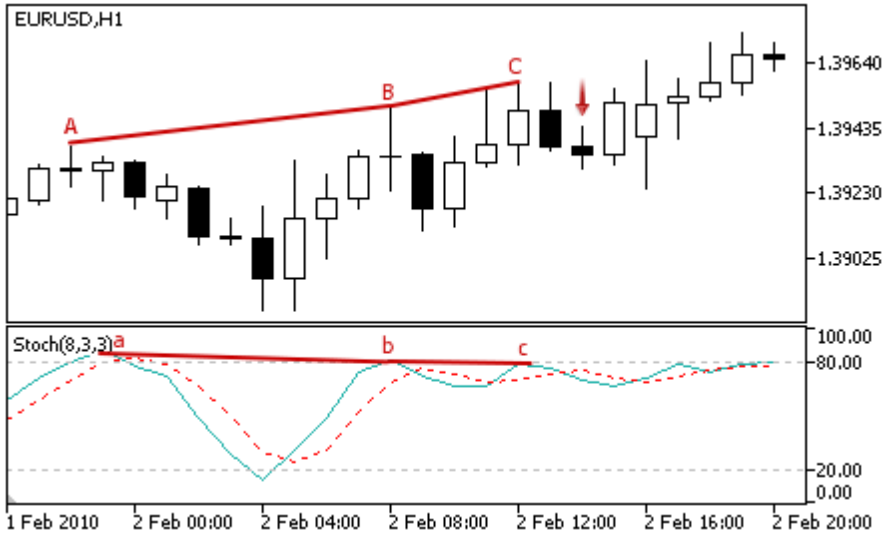
Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

| Signal Type | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| For buying  | <ul style="list-style-type: none"> <li> <b>Reverse</b> – the oscillator turned upwards (the oscillator rises at the analyzed bar and falls at the previous one).            </li> <li> <b>Crossing of the main and signal line</b> – the main line is above the signal line at the analyzed bar and below the signal line at the previous one.            </li> <li> <b>Divergence</b> – the first analyzed bottom of the oscillator is higher than the previous one, and the corresponding price bottom is lower than the previous one.            </li> </ul> |

| Signal Type | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|             |  <ul style="list-style-type: none"> <li>• <b>Double divergence</b> – the oscillator form three consequent bottoms, each of them is higher than the previous one; and the price formed three corresponding bottoms, and each of them is lower than the previous one.</li> </ul>  |
| For selling | <ul style="list-style-type: none"> <li>• <b>Reverse</b> – the oscillator turned downwards (the oscillator falls at the analyzed bar and rises at the previous one).</li> </ul>                                                                                                                                                                                                                                                                       |



| Signal Type | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|             | <div data-bbox="432 315 1321 846"> <p>EURUSD,H1</p> <p>Stoch(8,3,3)</p> </div> <ul style="list-style-type: none"> <li>• <b>Crossing of the main and signal line</b> – the main line is below the signal line at the analyzed bar and above the signal line at the previous one.</li> </ul> <div data-bbox="432 958 1321 1489"> <p>EURUSD,H1</p> <p>Stoch(8,3,3)</p> </div> <ul style="list-style-type: none"> <li>• <b>Divergence</b> – the first analyzed peak of the oscillator is lower than the previous one, and the corresponding price peak is higher than the previous peak.</li> </ul> |

| Signal Type              | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                           |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                          |  <p>• <b>Double divergence</b> – the oscillator formed three consequent peaks, each of them is lower than the previous one; and the price formed three corresponding peaks, each of them is higher than the previous one.</p>  |
| No objections to buying  | Value of the oscillator grows at the analyzed bar.                                                                                                                                                                                                                                                                                                                                                  |
| No objections to selling | Value of the oscillator falls at the analyzed bar.                                                                                                                                                                                                                                                                                                                                                  |

#### Note

Depending on the mode of operation of an Expert Advisor ("Every tick" or "Open prices only") an analyzed bar is either the current bar (with index 0), or the last formed bar (with index 1).

## Adjustable Parameters

This module has the following adjustable parameters:

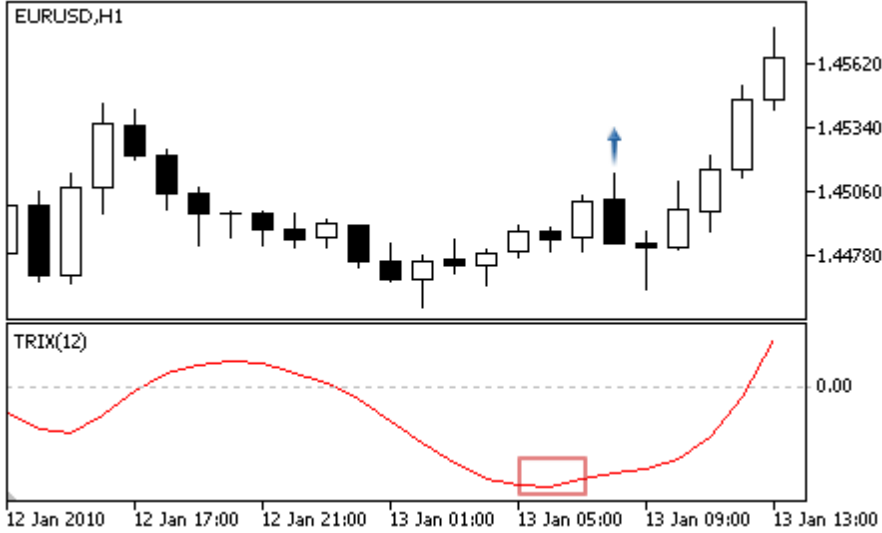
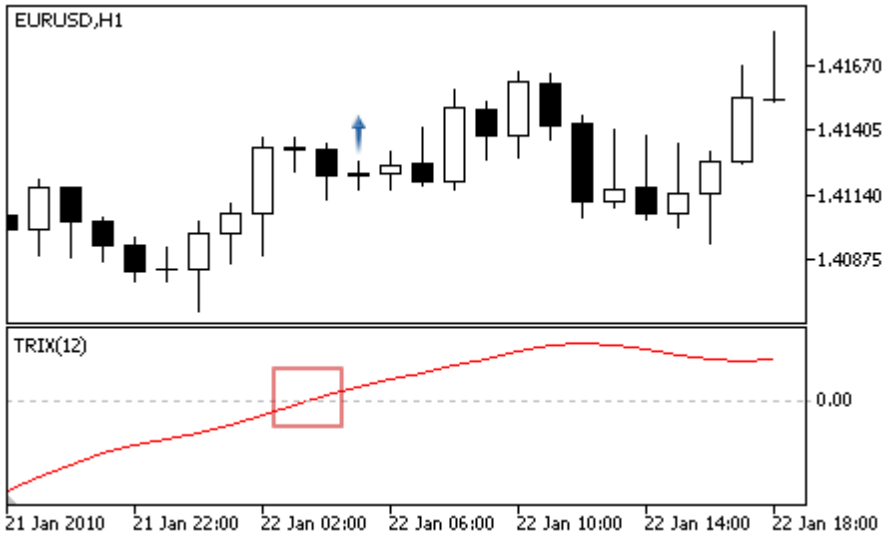
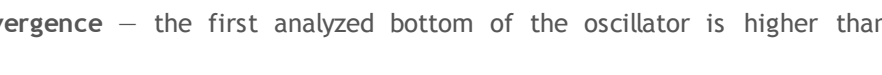
| Parameter  | Description                                                            |
|------------|------------------------------------------------------------------------|
| Weight     | Weight of signal of the module in the interval 0 to 1.                 |
| PeriodK    | Period of calculation of the main line of the oscillator.              |
| PeriodD    | Period of calculation of the main line of the oscillator.              |
| PeriodSlow | Period of slowing.                                                     |
| Applied    | A <a href="#">price series</a> used for calculation of the oscillator. |

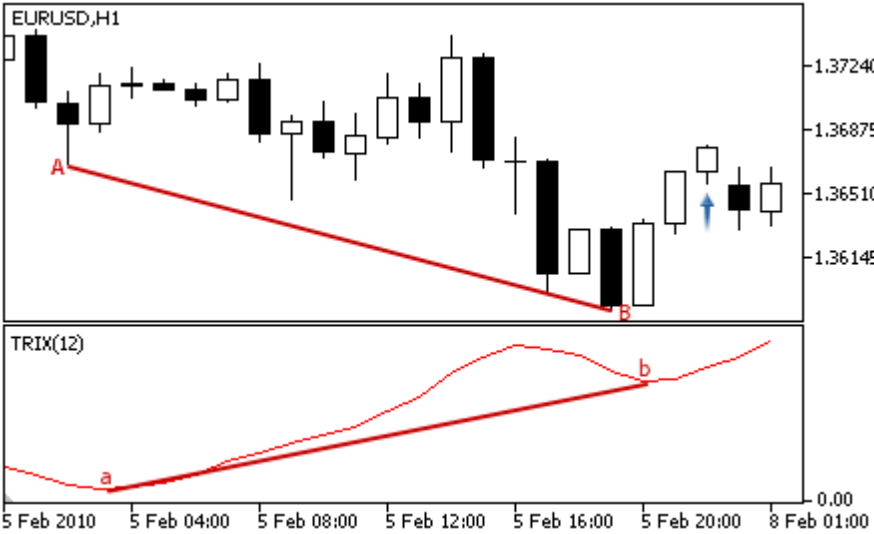
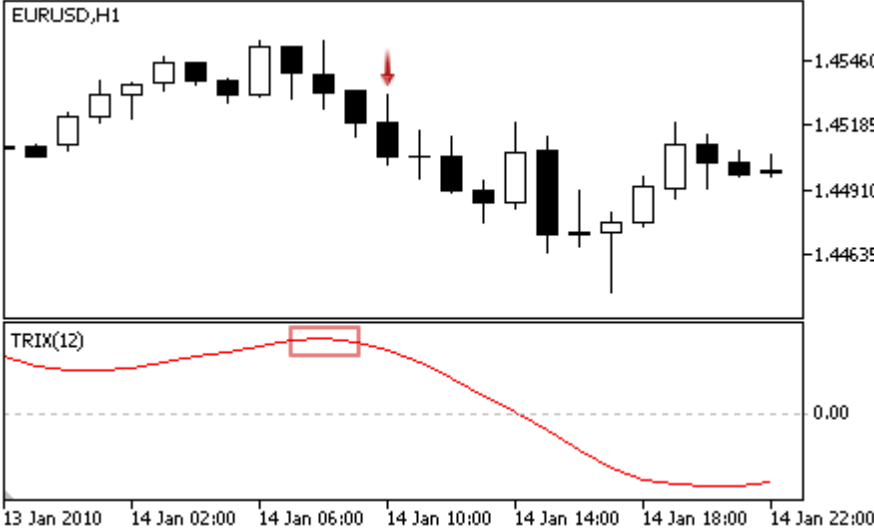
## Signals of the Oscillator Triple Exponential Average

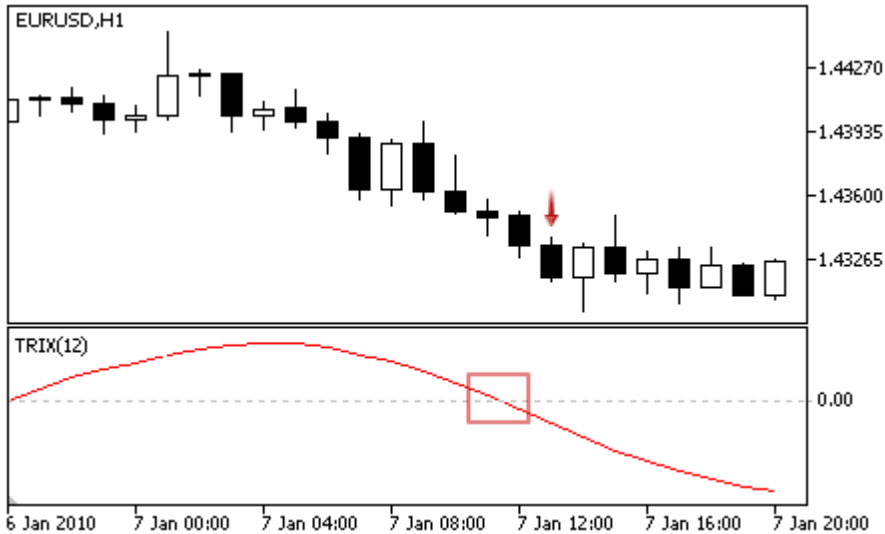
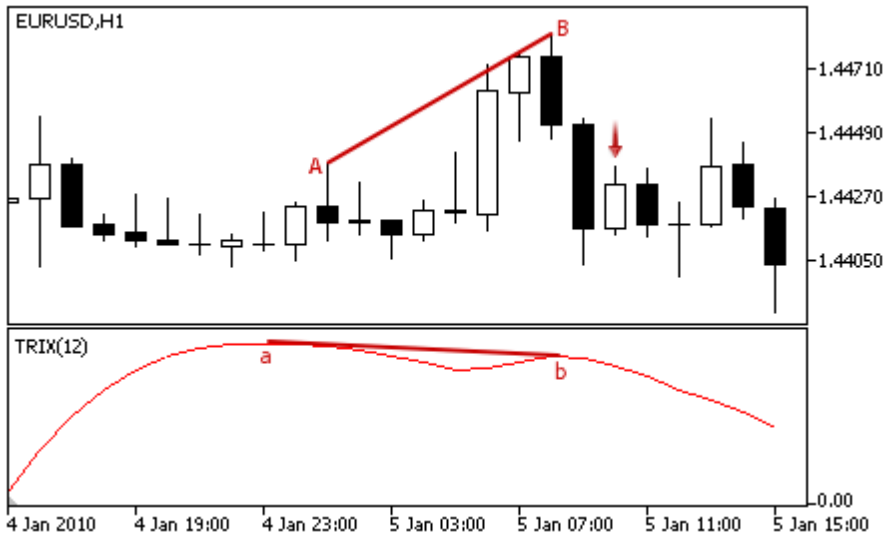
This module of signals is based on the market models of the oscillator [Triple Exponential Average](#). The mechanism of making trade decisions based on signals obtained from the modules is described in a [separate section](#).

### Conditions of Generation of Signals

Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

| Signal Type | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| For buying  | <ul style="list-style-type: none"> <li> <b>Reverse</b> – the oscillator turned upwards (the oscillator rises at the analyzed bar and falls at the previous one).            </li> <li> <b>Crossing the zero level</b> – the main line is above the zero level at the analyzed bar and below the zero level at the previous one.            </li> <li> <b>Divergence</b> – the first analyzed bottom of the oscillator is higher than the            </li> </ul> |

| Signal Type | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|             | <p>previous one, and the corresponding price bottom is lower than the previous one.</p>                                                                                                                                                                                                                                                                                       |
| For selling | <ul style="list-style-type: none"> <li>• <b>Reverse</b> – the oscillator turned downwards (the oscillator falls at the analyzed bar and rises at the previous one).</li> </ul>  <ul style="list-style-type: none"> <li>• <b>Crossing the zero level</b> – the main line is below the zero level at the analyzed bar and above the zero level at the previous one.</li> </ul> |

| Signal Type              | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                   |
|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                          |  <ul style="list-style-type: none"> <li>• <b>Divergence</b> – the first analyzed peak of the oscillator is lower than the previous one, and the corresponding price peak is higher than the previous peak.</li> </ul>  |
| No objections to buying  | Value of the oscillator grows at the analyzed bar.                                                                                                                                                                                                                                                                                                                                          |
| No objections to selling | Value of the oscillator falls at the analyzed bar.                                                                                                                                                                                                                                                                                                                                          |

#### Note

Depending on the mode of operation of an Expert Advisor ("Every tick" or "Open prices only") an analyzed bar is either the current bar (with index 0), or the last formed bar (with index 1).

## Adjustable Parameters

This module has the following adjustable parameters:

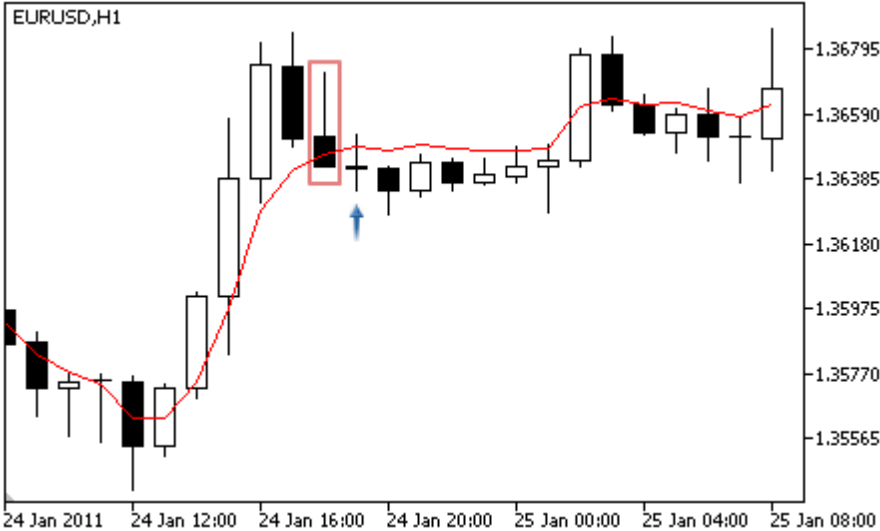
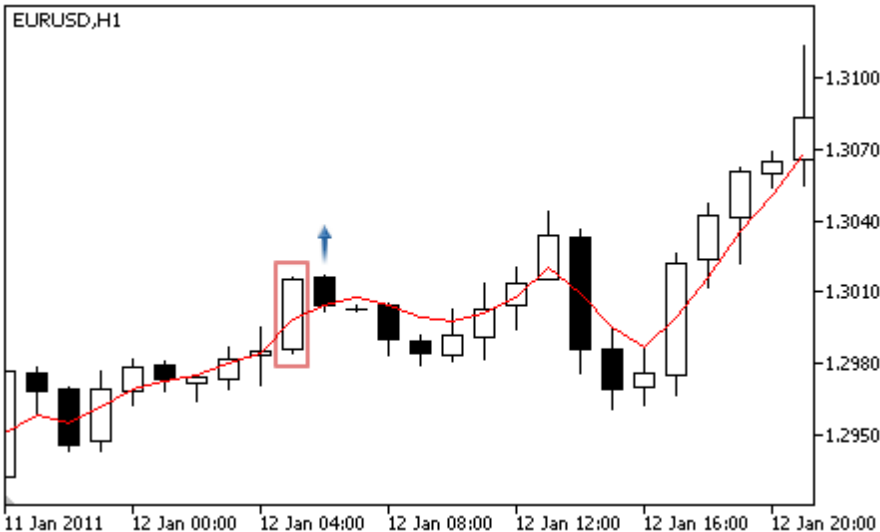
| Parameter  | Description                                                            |
|------------|------------------------------------------------------------------------|
| Weight     | Weight of signal of the module in the interval 0 to 1.                 |
| PeriodTriX | Period of calculation of the oscillator.                               |
| Applied    | A <a href="#">price series</a> used for calculation of the oscillator. |

## Signals of the Indicator Triple Exponential Moving Average

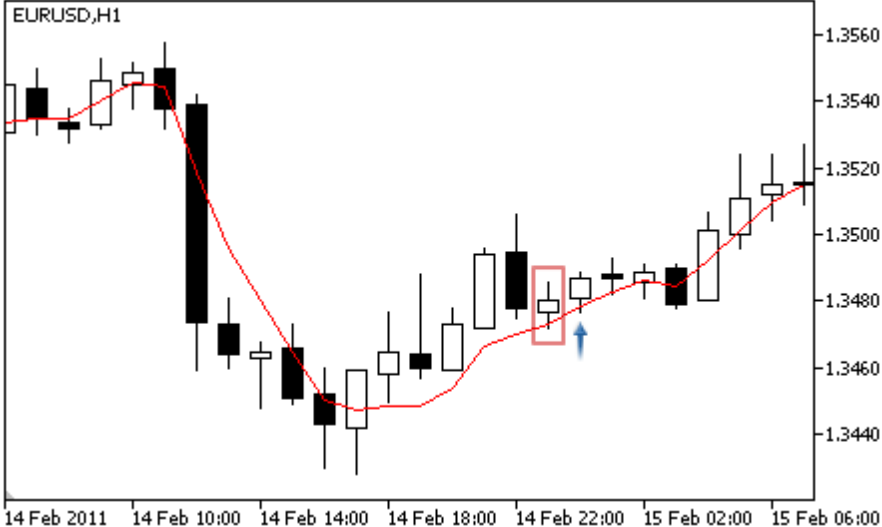
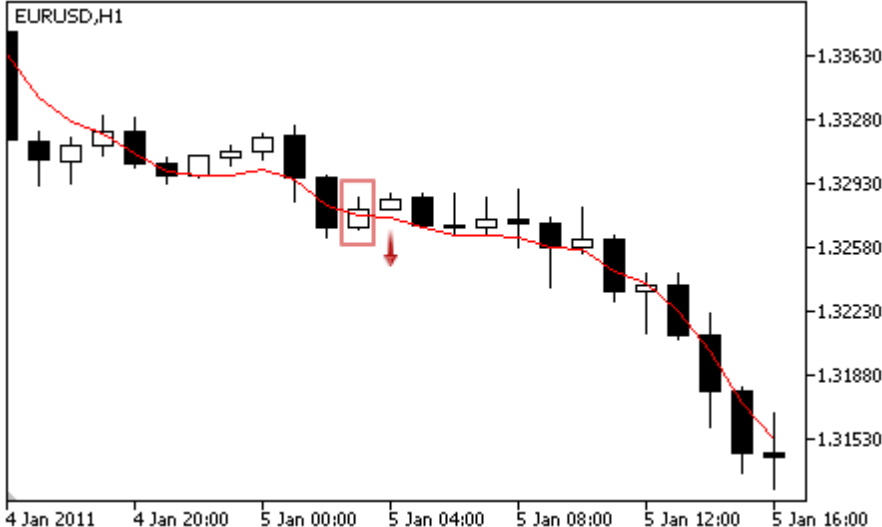
This module of signals is based on the market models of the indicator [Triple Exponential Moving Average](#). The mechanism of making trade decisions based on signals obtained from the modules is described in a [separate section](#).

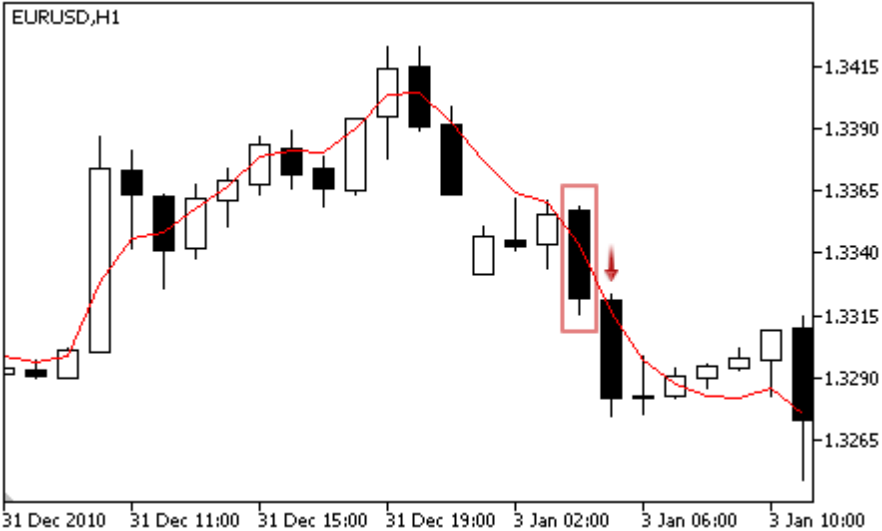
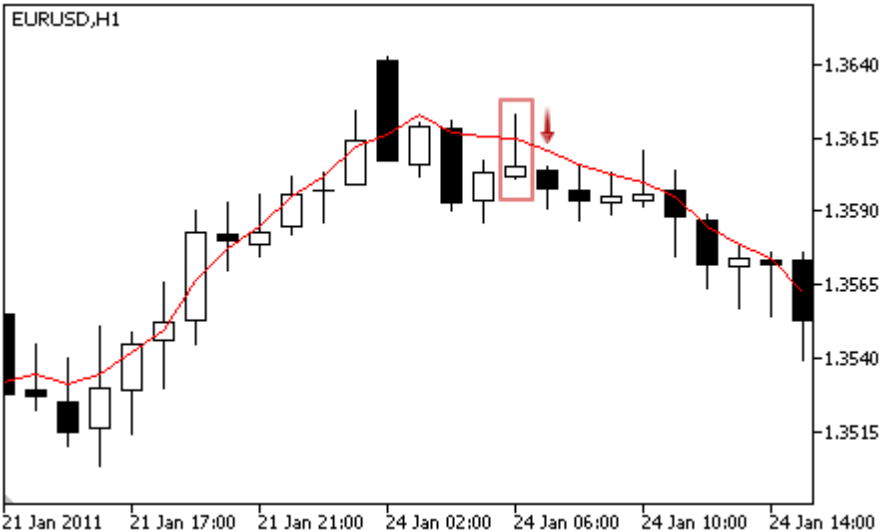
### Conditions of Generation of Signals

Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

| Signal Type | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| For buying  | <ul style="list-style-type: none"> <li>The price has crossed the indicator downwards (the Open price of the analyzed bar is above the indicator and the Close price is below the indicator) and the indicator rises (weak signal).</li> </ul>  <ul style="list-style-type: none"> <li><b>Moving Average crossover.</b> The price has crossed the indicator upwards (the Open price of the analyzed bar is below the indicator and the Close price is above the indicator) and the indicator rises (strong signal).</li> </ul>  |



| Signal Type | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|             | <ul style="list-style-type: none"> <li>The lower shadow of the bar has crossed the indicator (the Open and Close prices of the analyzed bar is above the indicator, and the Low price is below the indicator) and the indicator rises (weak signal).</li> </ul>                                                                                                                                                                                                                                                                 |
| For selling | <ul style="list-style-type: none"> <li>The price has crossed the indicator upwards (the Open price of the analyzed bar is below the indicator and the Close price is above the indicator) and the indicator falls (weak signal).</li> </ul>  <ul style="list-style-type: none"> <li><b>Moving Average crossover.</b> The price has crossed the indicator downwards (the Open price of the analyzed bar is above the indicator and the Close price is below the indicator) and the indicator falls (strong signal).</li> </ul> |

| Signal Type              | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                          | <div></div> <ul style="list-style-type: none"><li>• The upper shadow of the bar has crossed the indicator (the Open and Close prices of the analyzed bar is below the indicator, and the High price is above the indicator) and the indicator falls (weak signal).</li></ul> <div></div> |
| No objections to buying  | The prices is above the indicator.                                                                                                                                                                                                                                                                                                                                                                                                                            |
| No objections to selling | The price is below the indicator.                                                                                                                                                                                                                                                                                                                                                                                                                             |

Note

Depending on the mode of operation of an Expert Advisor ("Every tick" or "Open prices only") an analyzed bar is either the current bar (with index 0), or the last formed bar (with index 1).

Adjustable Parameters

This module has the following adjustable parameters:

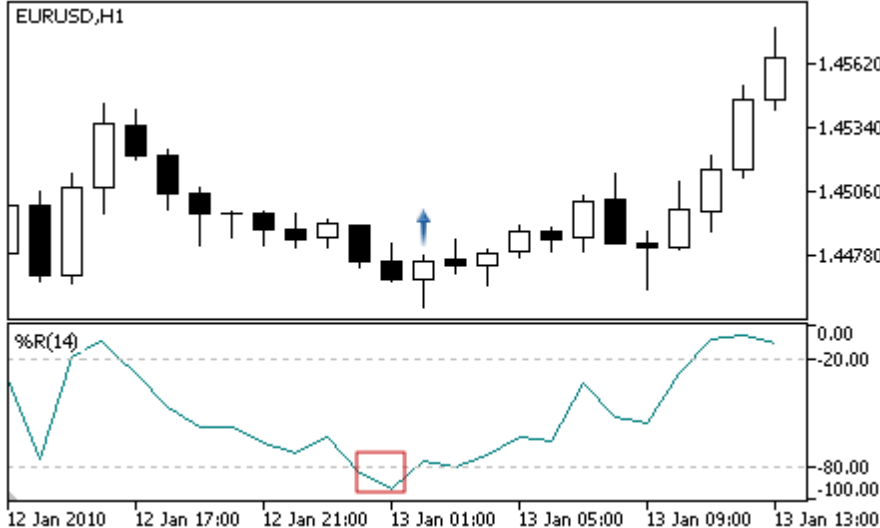
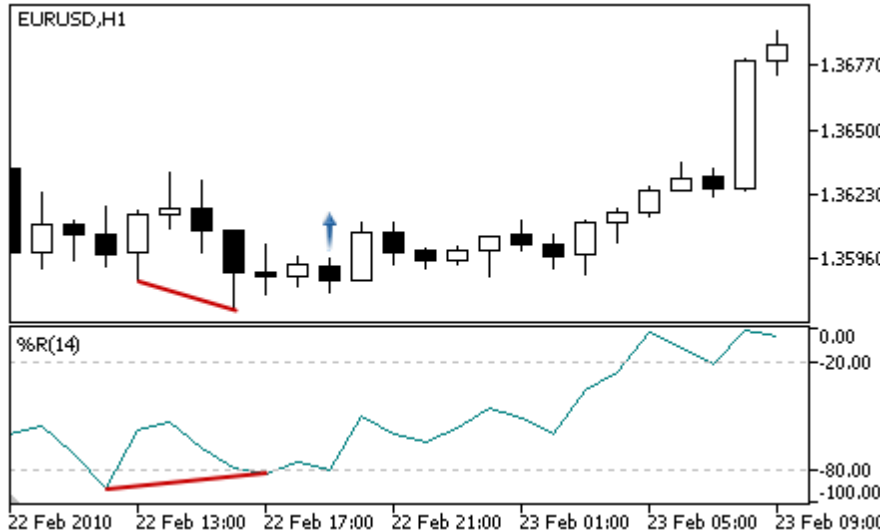
| Parameter | Description                                                           |
|-----------|-----------------------------------------------------------------------|
| Weight    | Weight of signal of the module in the interval 0 to 1.                |
| PeriodMA  | Period of averaging of the indicator.                                 |
| Shift     | Shift of the indicator along the time axis (in bars).                 |
| Method    | <a href="#">Method of averaging</a> .                                 |
| Applied   | A <a href="#">price series</a> used for calculation of the indicator. |

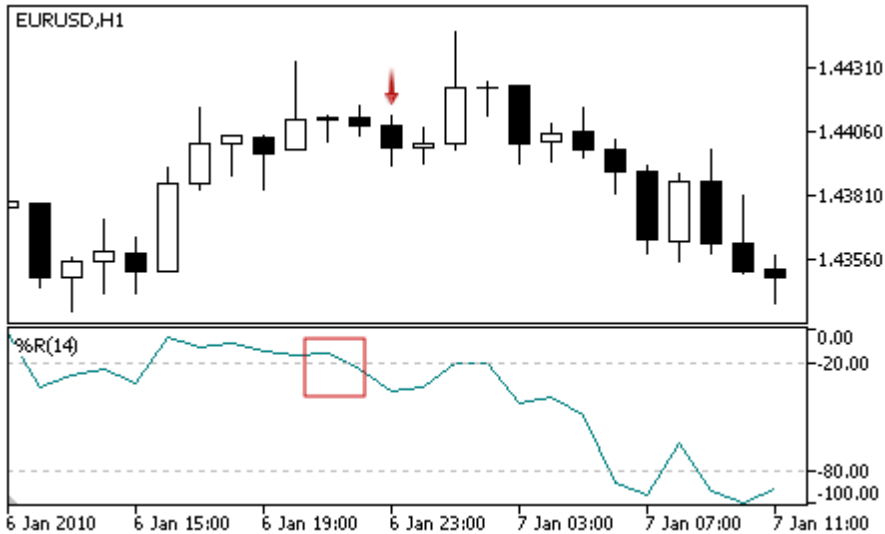
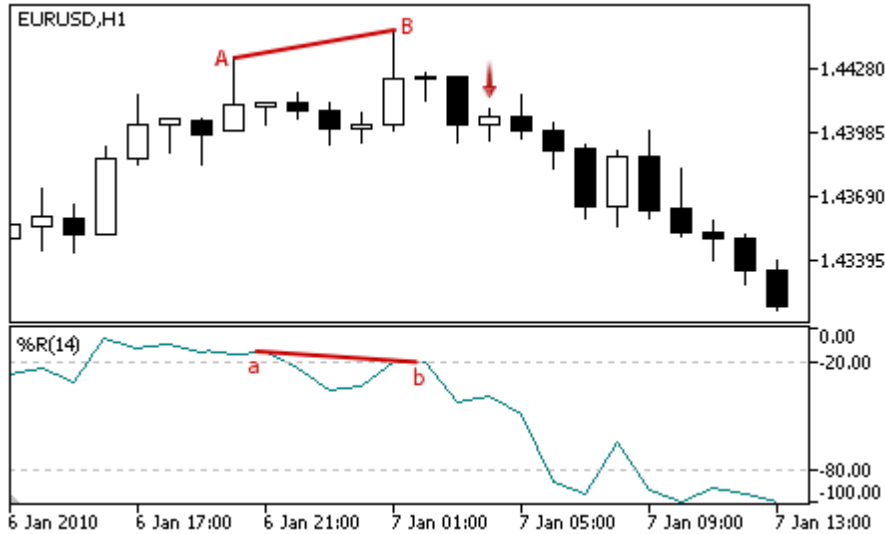
## Signals of the Oscillator Williams Percent Range

This module of signals is based on the market models of the oscillator [Williams Percent Range](#). The mechanism of making trade decisions based on signals obtained from the modules is described in a [separate section](#).

### Conditions of Generation of Signals

Below you can find the description of conditions when the module passes a signal to an Expert Advisor.

| Signal Type | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| For buying  | <ul style="list-style-type: none"> <li> <b>Reverse behind the oversold level</b> – the oscillator turned upwards and its value at the analyzed bar is behind the oversold level (default value is -20).            </li> <li> <b>Divergence</b> – the first analyzed bottom of the oscillator is higher than the previous one, and the corresponding price bottom is lower than the previous one.            </li> </ul> |
| For selling | <ul style="list-style-type: none"> <li> <b>Reverse behind the overbought level</b> – the oscillator turned downwards and its           </li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                              |

| Signal Type              | Description of Conditions                                                                                                                                                                                                                                                                                                                                                                                                                |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                          | <p>value at the analyzed bar is behind the overbought level (default value is -80).</p>  <p>• <b>Divergence</b> – the first analyzed peak of the oscillator is lower than the previous one, and the corresponding price peak is higher than the previous peak.</p>  |
| No objections to buying  | Value of the oscillator grows at the analyzed bar.                                                                                                                                                                                                                                                                                                                                                                                       |
| No objections to selling | Value of the oscillator falls at the analyzed bar.                                                                                                                                                                                                                                                                                                                                                                                       |

#### Note

Depending on the mode of operation of an Expert Advisor ("Every tick" or "Open prices only") an analyzed bar is either the current bar (with index 0), or the last formed bar (with index 1).

Remember that the oscillator Williams Percent Range has a reversed scale. Its maximum value is -100,

and minimum is 0.

## Adjustable Parameters

This module has the following adjustable parameters:

| Parameter | Description                                            |
|-----------|--------------------------------------------------------|
| Weight    | Weight of signal of the module in the interval 0 to 1. |
| PeriodWPR | Period of calculation of the oscillator.               |

## Trailing Stop classes

This section contains technical details of working with trailing stop classes and description of the relevant components of the MQL5 standard library.

The use of these classes will save time when creating (and testing) of trading strategies.

MQL5 Standard Library (in terms of trading strategies) is placed in the terminal directory, in the Include\Expert\Trailing folder.

| Class                                     | Description                                                                                    |
|-------------------------------------------|------------------------------------------------------------------------------------------------|
| <a href="#"><u>CTrailingFixedPips</u></a> | This class implements Trailing Stop algorithm, based on fixed points                           |
| <a href="#"><u>CTrailingMA</u></a>        | This class implements Trailing Stop algorithm, based on the values of Moving Average indicator |
| <a href="#"><u>CTrailingNone</u></a>      | A gag class, it doesn't uses any Trailing Stop algorithm                                       |
| <a href="#"><u>CTrailingPSAR</u></a>      | This class implements Trailing Stop algorithm, based on the values of Parabolic SAR indicator  |

## CTrailingFixedPips

CTrailingFixedPips is a class with implementation of Trailing Stop algorithm, based on fixed points trailing.

If position has Stop Loss price, it checks the minimal allowed Stop Loss distance to the current price. If its value is lower, that Stop Loss level, it suggests to set new Stop Loss price. For this case if position has Take Profit price, it suggests to set new Take Profit price.

If Expert Advisor has been [initialized](#) with the flag `every_tick=false`, it will perform all operations (trading, trailing, etc) only at the new bar. For this case Take profit level can be used. It will allow you to close opened position at Take Profit price before the new bar will be completed.

### Description

CTrailingFixedPips implements the Trailing Stop algorithm, based on positions trailing with the fixed points.

### Declaration

```
class CTrailingFixedPips: public CExpertTrailing
```

### Title

```
#include <Expert\Trailing\CTrailingFixedPips.mqh>
```

### Class Methods

| Initialization                                 |                                                  |
|------------------------------------------------|--------------------------------------------------|
| <a href="#">StopLevel</a>                      | Sets the value of Stop Loss level                |
| <a href="#">ProfitLevel</a>                    | Sets the value of Take Profit level              |
| virtual <a href="#">ValidationSettings</a>     | Checks the settings                              |
| <b>Check Trailing Methods</b>                  |                                                  |
| virtual <a href="#">CheckTrailingStopLong</a>  | Check Trailing Stop conditions of long position  |
| virtual <a href="#">CheckTrailingStopShort</a> | Check Trailing Stop conditions of short position |



## StopLevel

Sets the value Stop Loss level (in points).

```
void StopLevel(  
    int stop_level    // Stop Loss level  
)
```

### Parameters

*stop\_loss*

[in] The value of Stop Loss level (in conventional 2/4-digit points).

### Note

If Stop Loss level is equal to 0, the Trailing Stop is not used.

## ProfitLevel

Sets the value of Take Profit level (in points).

```
void ProfitLevel(  
    int profit_level    // Take profit level  
)
```

### Parameters

*profit\_level*

[in] The value of Take Profit level (in conventional 2/4-digit points).

### Note

If profit level is equal to 0, the Trailing Stop is not used.

## ValidationSettings

Checks the settings.

```
virtual bool ValidationSettings()
```

### Returned value

true if successful, otherwise false.

### Note

The function checks Take Profit and Stop Loss levels. The correct values are 0 and values, greater than minimal stop for stop orders for the symbol.

## CheckTrailingStopLong

Checks Trailing Stop conditions of long position.

```
virtual bool CheckTrailingStopLong(  
    CPositionInfo* position, // CPositionInfo object pointer  
    double& sl, // Stop Loss price  
    double& tp // Take Profit price  
)
```

### Parameters

*position*

[in] Pointer to [CPositionInfo](#) object.

*sl*

[in][out] Variable for Stop Loss price.

*tp*

[in][out] Variable for Take Profit price.

### Returned value

true if conditions are satisfied, otherwise false.

### Note

If Stop Loss level is equal to 0, the Trailing Stop is not used. If position already has Stop Loss price, its value is assumed as a base price, otherwise the position open price is assumed as a base price.

If the current Bid price is higher than base price+stop loss level, it suggests to set new Stop Loss price. In this case, If position already has Take Profit price, it suggests to set new Take Profit price equal to Bid price+take profit level.

## CheckTrailingStopShort

Checks Trailing Stop conditions of short position.

```
virtual bool CheckTrailingStopShort (
    CPositionInfo* position,      // CPositionInfo object pointer
    double& sl,                  // Stop Loss price
    double& tp                    // Take Profit price
)
```

### Parameters

*position*

[in] Pointer to [CPositionInfo](#) object.

*sl*

[in][out] Variable for Stop Loss price.

*tp*

[in][out] Variable for Take Profit price.

### Returned value

true if conditions are satisfied, otherwise false.

### Note

If Stop Loss level is equal to 0, the Trailing Stop is not used. If position already has Stop Loss price, its value is assumed as a base price, otherwise the position open price is assumed as a base price.

If the current Ask price is lower than base price-stop loss level, it suggests to set new Stop Loss price. In this case, If position already has Take Profit price, it suggests to set new Take Profit price equal to Ask price-take profit level.

## CTrailingMA

CTrailingMA is a class with implementation of Trailing Stop algorithm, based on the values of moving average indicator.

### Description

CTrailingMA class implements Trailing Stop algorithm, based on the values of moving average indicator of the previous (completed) bar.

### Declaration

```
class CTrailingMA: public CExpertTrailing
```

### Title

```
#include <Expert\Trailing\TrailingMA.mqh>
```

### Class Methods

| Initialization                        |                                                  |
|---------------------------------------|--------------------------------------------------|
| <u>Period</u>                         | Sets period of moving average                    |
| <u>Shift</u>                          | Sets shift of moving average                     |
| <u>Method</u>                         | Sets smoothing method of moving average          |
| <u>Applied</u>                        | Sets applied price of moving average             |
| virtual <u>InitIndicators</u>         | Initializes indicators and time series           |
| virtual <u>ValidationSettings</u>     | Checks the settings                              |
| <b>Check Trailing Methods</b>         |                                                  |
| virtual <u>CheckTrailingStopLong</u>  | Check Trailing Stop conditions of long position  |
| virtual <u>CheckTrailingStopShort</u> | Check Trailing Stop conditions of short position |

## Period

Sets period of moving average.

```
void Period(  
    int period    // Smoothing period  
)
```

### Parameters

*period*

[in] Period of moving average.

## Shift

Sets shift of moving average.

```
void Shift(  
    int shift    // Shift  
)
```

### Parameters

*shift*

[in] Shift of moving average.



## Method

Sets smoothing method of moving average.

```
void Method(  
    ENUM_MA_METHOD method      // Smoothing method  
)
```

### Parameters

*method*

[in] Smoothing method of moving average indicator.

## Applied

Sets applied price of moving average.

```
void Applied(  
    ENUM_APPLIED_PRICE applied // Applied price  
)
```

### Parameters

*applied*

[in] Applied price of moving average.

## InitIndicators

Initializes indicators and time series.

```
virtual bool InitIndicators(  
    CIndicators* indicators // CIndicators collection pointer  
)
```

### Parameters

*indicators*

[in] Pointer to indicators and time series collection ([CExpert](#) class member).

### Returned value

true if successful, otherwise false.

## ValidationSettings

Checks the settings.

```
virtual bool ValidationSettings()
```

### Returned value

true if successful, otherwise false.

### Note

The function checks the period of moving average, the correct values are positive.

## CheckTrailingStopLong

Checks Trailing Stop conditions of long position.

```
virtual bool CheckTrailingStopLong(  
    CPositionInfo* position,    // CPositionInfo object pointer  
    double& sl,                // Stop Loss price  
    double& tp                 // Take Profit price  
)
```

### Parameters

*position*

[in] Pointer to [CPositionInfo](#) object.

*sl*

[in][out] Variable for Stop Loss price.

*tp*

[in][out] Variable for Take Profit price.

### Returned value

true if conditions are satisfied, otherwise false.

### Note

The first it calculates the maximal allowed Stop Loss price, closest to the current price and calculates Stop Loss price using the values of moving average indicator of the previous (completed) bar.

If position already has Stop Loss price, its value is assumed as a base price, otherwise the base price is the open price of the position.

If the calculated Stop Loss price is higher than base price and lower than maximal allowed Stop Loss price, it suggests to set new Stop Loss price.

## CheckTrailingStopShort

Checks Trailing Stop conditions of short position.

```
virtual bool CheckTrailingStopShort (
    CPositionInfo* position,      // CPositionInfo object pointer
    double& sl,                  // Stop Loss price
    double& tp                   // Take Profit price
)
```

### Parameters

*position*

[in] Pointer to [CPositionInfo](#) object.

*sl*

[in][out] Variable for Stop Loss price.

*tp*

[in][out] Variable for Take Profit price.

### Returned value

true if conditions are satisfied, otherwise false.

### Note

The first it calculates the minimal allowed Stop Loss price, closest to the current price and calculates Stop Loss price using the values of moving average indicator of the previous (completed) bar.

If position already has Stop Loss price, its value is assumed as a base price, otherwise the base price is the open price of the position.

If the calculated Stop Loss price is higher than base price and lower than minimal allowed Stop Loss price, it suggests to set new Stop Loss price.

## CTrailingNone

CTrailingNone is a gag class. This class should be used at initialization of Trailing object if your strategy doesn't use Trailing Stop.

### Description

CTrailingNone class doesn't implement any Trailing Stop algorithms. The methods of checking Trailing Stop conditions always return false.

### Declaration

```
class CTrailingNone: public CExpertTrailing
```

### Title

```
#include <Expert\Trailing\TrailingNone.mqh>
```

### Class Methods

| Check Trailing Methods                         |                                                                   |
|------------------------------------------------|-------------------------------------------------------------------|
| virtual <a href="#">CheckTrailingStopLong</a>  | A gag method for check Trailing Stop conditions of long position  |
| virtual <a href="#">CheckTrailingStopShort</a> | A gag method for check Trailing Stop conditions of short position |

## CheckTrailingStopLong

Checks Trailing Stop conditions of long position.

```
virtual bool CheckTrailingStopLong(  
    CPositionInfo* position,      // CPositionInfo object pointer  
    double& sl,                  // Stop Loss price  
    double& tp                    // Take Profit price  
)
```

### Parameters

*position*

[in] Pointer to [CPositionInfo](#) object.

*sl*

[in][out] Variable for Stop Loss price.

*tp*

[in][out] Variable for Take Profit price.

### Returned value

true if conditions are satisfied, otherwise false.

### Note

The function always returns false.



## CheckTrailingStopShort

Checks Trailing Stop conditions of short position.

```
virtual bool CheckTrailingStopShort (  
    CPositionInfo* position,      // CPositionInfo object pointer  
    double& sl,                  // Stop Loss price  
    double& tp                   // Take Profit price  
)
```

### Parameters

*position*

[in] Pointer to [CPositionInfo](#) object.

*sl*

[in][out] Variable for Stop Loss price.

*tp*

[in][out] Variable for Take Profit price.

### Returned value

true if conditions are satisfied, otherwise false.

### Note

The function always returns false.

## CTrailingPSAR

CTrailingPSAR is a class with implementation of Trailing Stop algorithm, based on the values of Parabolic SAR indicator.

### Description

CTrailingPSAR class implements the Trailing Stop algorithm, based on the values of Parabolic SAR indicator of the previous (completed) bar.

### Declaration

```
class CTrailingPSAR: public CExpertTrailing
```

### Title

```
#include <Expert\Trailing\TrailingParabolicSAR.mqh>
```

### Class Methods

| Initialization                        |                                                      |
|---------------------------------------|------------------------------------------------------|
| <u>Step</u>                           | Sets the value of step of Parabolic SAR indicator    |
| <u>Maximum</u>                        | Sets the value of maximum of Parabolic SAR indicator |
| virtual <u>InitIndicators</u>         | Initializes indicators and time series               |
| <b>Check Trailing Methods</b>         |                                                      |
| virtual <u>CheckTrailingStopLong</u>  | Check conditions of trailing stop of long position   |
| virtual <u>CheckTrailingStopShort</u> | Check conditions of trailing stop of short position  |

## Step

Sets the value of step of Parabolic SAR indicator.

```
void Step(  
    double step    // Step  
)
```

### Parameters

*step*

[in] The value of Step of Parabolic SAR indicator.

## Maximum

Sets the value of maximum of Parabolic SAR indicator.

```
void Maximum(  
    double maximum    // Maximum  
)
```

### Parameters

*maximum*

[in] The value of maximum of Parabolic SAR indicator.

## InitIndicators

Initializes indicators and time series.

```
virtual bool InitIndicators(  
    CIndicators* indicators // CIndicators collection pointer  
)
```

### Parameters

*indicators*

[in] Pointer to indicators and time series collection ([CExpert](#) class member).

### Returned value

true if successful, otherwise false.

## CheckTrailingStopLong

Checks Trailing Stop conditions of long position.

```
virtual bool CheckTrailingStopLong(  
    CPositionInfo* position,    // Pointer  
    double& sl,                // Link  
    double& tp                 // Link  
)
```

### Parameters

*position*

[in] Pointer to [CPositionInfo](#) object.

*sl*

[in][out] Variable for Stop Loss price.

*tp*

[in][out] Variable for Take Profit price.

### Returned value

true if conditions are satisfied, otherwise false.

### Note

The first it calculates the maximal allowed Stop Loss price, closest to the current price and calculates Stop Loss price using the values of Parabolic SAR indicator of the previous (completed) bar.

If position already has Stop Loss price, its value is assumed as a base price, otherwise the position open price is assumed as a base price.

If the calculated Stop Loss price is higher than base price and lower than maximal allowed Stop Loss price, it suggests to set new Stop Loss price.

## CheckTrailingStopShort

Checks Trailing Stop conditions of short position.

```
virtual bool CheckTrailingStopShort (
    CPositionInfo* position,    // Pointer
    double& sl,                // Link
    double& tp                 // Link
)
```

### Parameters

*position*

[in] Pointer to [CPositionInfo](#) object.

*sl*

[in][out] Variable for Stop Loss price.

*tp*

[in][out] Variable for Take Profit price.

### Returned value

true if conditions are satisfied, otherwise false.

### Note

The first it calculates the minimal allowed Stop Loss price, closest to the current price and calculates Stop Loss price using the values of Parabolic SAR indicator of the previous (completed) bar.

If position already has Stop Loss price, its value is assumed as a base price, otherwise the position open price is assumed as a base price.

If the calculated Stop Loss price is higher than base price and lower than minimal allowed Stop Loss price, it suggests to set new Stop Loss price.

## Money Management classes

This section contains technical details of working with money and risk management classes and description of the relevant components of the MQL5 standard library.

The use of these classes will save time when creating (and testing) of trading strategies.

MQL5 Standard Library (in terms of money and risk management classes) is placed in the terminal directory, in the Include\Expert\Money\ folder.

| Class                                      | Description                                                                                                                            |
|--------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#"><u>CMoneyFixedLot</u></a>      | This class implements money management algorithm, based on trading with predefined fixed lot size.                                     |
| <a href="#"><u>CMoneyFixedMargin</u></a>   | This class implements money management algorithm, based on trading with predefined fixed margin.                                       |
| <a href="#"><u>CMoneyFixedRisk</u></a>     | This class implements money management algorithm, based on trading with predefined risk.                                               |
| <a href="#"><u>CMoneyNone</u></a>          | This class implements money management algorithm, based on trading with minimal allowed lot size.                                      |
| <a href="#"><u>CMoneySizeOptimized</u></a> | This class implements money management algorithm, based on trading with variable lot size, depending on results of the previous deals. |



## CMoneyFixedLot

CMoneyFixedLot is the class money management algorithm, based on trading with predefined fixed lot size.

### Description

CMoneyFixedLot implements money management algorithm, based on trading with predefined fixed lot size.

### Declaration

```
class CMoneyFixedLot: public CExpertMoney
```

### Title

```
#include <Expert\Money\MoneyFixedLot.mqh>
```

### Class Methods

| Initialization                             |                                      |
|--------------------------------------------|--------------------------------------|
| <a href="#">Lots</a>                       | Sets trading volume                  |
| virtual <a href="#">ValidationSettings</a> | Checks the settings                  |
| <b>Money and Risk Management Methods</b>   |                                      |
| virtual <a href="#">CheckOpenLong</a>      | Gets trade volume for long position  |
| virtual <a href="#">CheckOpenShort</a>     | Gets trade volume for short position |

## Lots

Sets trading volume (in lots).

```
void Lots(  
    double lots    // Lots  
)
```

### Parameters

*lots*

[in] Trading volume (in lots).

## ValidationSettings

Checks the settings.

```
virtual bool ValidationSettings()
```

### Returned value

true if successful, otherwise false.

### Note

Checks the specified trading volume for correctness.

## CheckOpenLong

Gets trade volume for long position.

```
virtual double CheckOpenLong(  
    double price,      // Price  
    double sl          // Stop Loss price  
)
```

### Parameters

*price*

[in] Price.

*sl*

[in] Stop Loss price.

### Returned value

Trade volume for long position.

### Note

The function always returns the fixed trade volume, defined by [Lots](#) method.

## CheckOpenShort

Gets trade volume for short position.

```
virtual double CheckOpenShort(  
    double price,      // Price  
    double sl          // Stop Loss price  
)
```

### Parameters

*price*

[in] Price.

*sl*

[in] Stop Loss price.

### Returned value

Trade volume for short position.

### Note

The function always returns the fixed trade volume, defined by [Lots](#) method.

## CMoneyFixedMargin

CMoneyFixedMargin is the class money management algorithm, based on trading with predefined fixed margin.

### Description

CMoneyFixedMargin implements money management algorithm, based on trading with predefined fixed margin.

### Declaration

```
class CMoneyFixedMargin: public CExpertMoney
```

### Title

```
#include <Expert\Money\MoneyFixedMargin.mqh>
```

### Class Methods

| Money and Risk Management Methods      |                                      |
|----------------------------------------|--------------------------------------|
| virtual <a href="#">CheckOpenLong</a>  | Gets trade volume for long position  |
| virtual <a href="#">CheckOpenShort</a> | Gets trade volume for short position |

## CheckOpenLong

Gets trade volume for long position.

```
virtual double CheckOpenLong(  
    double price,      // Price  
    double sl          // Stop Loss price  
)
```

### Parameters

*price*

[in] Price.

*sl*

[in] Stop Loss price.

### Returned value

Trade volume for long position.

### Note

The function returns trade volume for long position, it uses the fixed margin. The margin is defined by Percent parameter of [CExpertMoney](#) base class.

## CheckOpenShort

Gets trade volume for short position.

```
virtual double CheckOpenShort(  
    double price,      // Price  
    double sl          // Stop Loss price  
)
```

### Parameters

*price*

[in] Price.

*sl*

[in] Stop Loss price.

### Returned value

Trade volume for short position.

### Note

The function returns trade volume for short position, it uses the fixed margin. The margin is defined by Percent parameter of [CExpertMoney](#) base class.



## CMoneyFixedRisk

CMoneyFixedRisk is a class with implementation of money management algorithm with fixed predefined risk.

### Description

CMoneyFixedRisk class implements the money management algorithm with fixed predefined risk.

### Declaration

```
class CMoneyFixedRisk: public CExpertMoney
```

### Title

```
#include <Expert\Money\MoneyFixedRisk.mqh>
```

### Class Methods

| Money and Risk Management Methods      |                                      |
|----------------------------------------|--------------------------------------|
| virtual <a href="#">CheckOpenLong</a>  | Gets trade volume for long position  |
| virtual <a href="#">CheckOpenShort</a> | Gets trade volume for short position |

## CheckOpenLong

Gets trade volume for long position.

```
virtual double CheckOpenLong(  
    double price,      // Price  
    double sl          // Stop Loss price  
)
```

### Parameters

*price*

[in] Price.

*sl*

[in] Stop Loss price.

### Returned value

Trade volume for long position.

### Note

The function returns trade volume for long position, it uses the fixed risk. The risk is defined by Percent parameter of [CExpertMoney](#) base class.

## CheckOpenShort

Gets trade volume for short position.

```
virtual double CheckOpenShort(  
    double price,      // Price  
    double sl          // Stop Loss price  
)
```

### Parameters

*price*

[in] Price.

*sl*

[in] Stop Loss price.

### Returned value

Trade volume for short position.

### Note

The function returns trade volume for short position, it uses the fixed risk. The risk is defined by Percent parameter of [CExpertMoney](#) base class.

## CMoneyNone

CMoneyNone is a class with implementation of trading algorithm with minimal allowed lot.

### Description

CMoneyNone class implements trading with minimal allowed lot.

### Declaration

```
class CMoneyNone: public CExpertMoney
```

### Title

```
#include <Expert\Money\MoneyNone.mqh>
```

### Class Methods

| Initialization                             |                                      |
|--------------------------------------------|--------------------------------------|
| virtual <a href="#">ValidationSettings</a> | Checks the settings                  |
| Money and Risk Management Methods          |                                      |
| virtual <a href="#">CheckOpenLong</a>      | Gets trade volume for long position  |
| virtual <a href="#">CheckOpenShort</a>     | Gets trade volume for short position |

## ValidationSettings

Checks the settings.

```
virtual bool ValidationSettings()
```

### Returned value

true if successful, otherwise false.

### Note

The function always returns true.

## CheckOpenLong

Gets trade volume for long position.

```
virtual double CheckOpenLong(  
    double price,      // Price  
    double sl          // Stop Loss price  
)
```

### Parameters

*price*

[in] Price.

*sl*

[in] Stop Loss price.

### Returned value

Trade volume for long position.

### Note

The function always returns the minimal lot size.

## CheckOpenShort

Gets trade volume for long position.

```
virtual double CheckOpenShort(  
    double price,      // Price  
    double sl          // Stop Loss price  
)
```

### Parameters

*price*

[in] Price.

*sl*

[in] Stop Loss price.

### Returned value

Trade volume for short position.

### Note

The function always returns the minimal lot size.

# CMoneySizeOptimized

CMoneySizeOptimized is a class with implementation of money management algorithm, based on trading with variable lot size, depending on results of the previous deals.

## Description

CMoneySizeOptimized implements money management algorithm, based on trading with variable lot size, depending on results of the previous deals.

## Declaration

```
class CMoneySizeOptimized: public CExpertMoney
```

## Title

```
#include <Expert\Money\MoneySizeOptimized.mqh>
```

## Class Methods

|                                            |                                      |
|--------------------------------------------|--------------------------------------|
| Initialization                             |                                      |
| <a href="#">DecreaseFactor</a>             | Sets the value of decrease factor    |
| virtual <a href="#">ValidationSettings</a> | Checks the settings                  |
| Money and Risk Management Methods          |                                      |
| virtual <a href="#">CheckOpenLong</a>      | Gets trade volume for long position  |
| virtual <a href="#">CheckOpenShort</a>     | Gets trade volume for short position |



## DecreaseFactor

Sets the value of decrease factor.

```
void DecreaseFactor(  
    double decrease_factor    // Decrease factor  
)
```

### Parameters

*decrease\_factor*

[in] Decrease factor.

### Note

The DecreaseFactor defines the volume decreasing coefficient (compared with the volume of previous position) for the case of consecutive loss trades.

## ValidationSettings

Checks the settings.

```
virtual bool ValidationSettings()
```

### Returned value

true if successful, otherwise false.

### Note

If the value of decrease factor is negative, it returns false, otherwise it returns true.

## CheckOpenLong

Gets trade volume for long position.

```
virtual double CheckOpenLong(  
    double price,      // Price  
    double sl          // Stop Loss price  
)
```

### Parameters

*price*

[in] Price.

*sl*

[in] Stop Loss price.

### Returned value

Trade volume for long position.

### Note

The function returns trade volume for long position, the volume dependent on results of the previous deals.

## CheckOpenShort

Gets trade volume for short position.

```
virtual double CheckOpenShort(  
    double price,      // Price  
    double sl          // Stop Loss price  
)
```

### Parameters

*price*

[in] Price.

*sl*

[in] Stop Loss price.

### Returned value

Trade volume for long position.

### Note

The function returns trade volume for short position, the volume dependent on results of the previous deals.

## Classes for Creation of Control Panels and Dialogs

This section contains technical details of working with classes for creation of controls panels and description of the relevant components of the MQL5 standard library.

The use of these classes will save time when creating the control panels for MQL5 programs (Expert Advisors and indicators).

MQL5 Standard Library (in terms of controls) is placed in the client terminal data folder, in the MQL5 \Include\Controls.

The example of Expert Advisor, which illustrates the working of these classes can be found in MQL5 \Expert\Examples\Controls.

| Auxiliary classes             | Description                                 |
|-------------------------------|---------------------------------------------|
| <a href="#"><u>CPoint</u></a> | Class of the point in Cartesian coordinates |
| <a href="#"><u>CRect</u></a>  | Class of the rectangular area               |

| Base classes                         | Description                                                     |
|--------------------------------------|-----------------------------------------------------------------|
| <a href="#"><u>CWnd</u></a>          | Base class for all controls                                     |
| <a href="#"><u>CWndObj</u></a>       | Base class for controls and dialogs                             |
| <a href="#"><u>CWndContainer</u></a> | Base class for complex controls (containing dependent controls) |

| Simple controls                   | Description                                     |
|-----------------------------------|-------------------------------------------------|
| <a href="#"><u>CLabel</u></a>     | Control, based on "Text label" graphic object   |
| <a href="#"><u>CBmpButton</u></a> | Control, based on "Bitmap label" graphic object |
| <a href="#"><u>CButton</u></a>    | Control, based on "Button" graphic object       |
| <a href="#"><u>CEdit</u></a>      | Control, based on "Edit field" graphic object   |
| <a href="#"><u>CPanel</u></a>     | Control, based on "Rectangle label"             |
| <a href="#"><u>CPicture</u></a>   | Control, based on "Bitmap label"                |

| Complex controls                  | Description                                    |
|-----------------------------------|------------------------------------------------|
| <a href="#"><u>CScroll</u></a>    | Base class of the scroll bar                   |
| <a href="#"><u>CScrollV</u></a>   | Vertical scroll bar                            |
| <a href="#"><u>CScrollH</u></a>   | Horizontal scroll bar                          |
| <a href="#"><u>CWndClient</u></a> | Base class of the client area with scroll bars |
| <a href="#"><u>CListView</u></a>  | ListView                                       |
| <a href="#"><u>CComboBox</u></a>  | ComboBox                                       |

|                                     |                    |
|-------------------------------------|--------------------|
| <a href="#"><u>CCheckBox</u></a>    | CheckBox           |
| <a href="#"><u>CCheckGroup</u></a>  | CheckGroup         |
| <a href="#"><u>CRadioButton</u></a> | RadioButton        |
| <a href="#"><u>CRadioGroup</u></a>  | RadioGroup         |
| <a href="#"><u>CSpinEdit</u></a>    | SpinEdit           |
| <a href="#"><u>CDialog</u></a>      | Dialog             |
| <a href="#"><u>CAppDialog</u></a>   | Application Dialog |

## CPoint

CPoint is a class of the point.

### Description

CPoint is a class of the point in Cartesian coordinates.

### Declaration

```
class CPoint
```

### Title

```
#include <Controls\Rect.mqh>
```

### Class Methods

| Geometry               |                                                      |
|------------------------|------------------------------------------------------|
| <a href="#">Move</a>   | Sets new coordinates of the point                    |
| <a href="#">Shift</a>  | Performs the relative shift of the point coordinates |
| Additional methods     |                                                      |
| <a href="#">Format</a> | Gets the point coordinates as string                 |

## Move

Sets new coordinates of the point.

```
void Move(  
    const int x,      // X coordinate  
    const int y      // Y coordinate  
)
```

### Parameters

*x*

[in] New X coordinate.

*y*

[in] New Y coordinate.

### Returned value

None.



## Shift

Performs the relative shift of the point coordinates.

```
void Shift(  
    const int dx,      // delta X  
    const int dy      // delta Y  
)
```

### Parameters

*dx*

[in] Delta X.

*dy*

[in] Delta Y.

### Returned value

None.

## Format

Gets the point coordinates as string.

```
string Format(  
    string & fmt,      // format  
    ) const
```

### Parameters

*fmt*

[in] String with format.

### Returned value

String with the point coordinates.

## CRect

CRect is a class of the rectangular area of the chart.

### Description

CRect is a class of the area, it defined by coordinates of the upper-left and lower-right corners of a rectangle in Cartesian coordinates.

### Declaration

```
class CRect
```

### Title

```
#include <Controls\Rect.mqh>
```

### Class Methods

| Properties                |                                                      |
|---------------------------|------------------------------------------------------|
| <u>Left</u>               | Gets/Sets the X coordinate of the upper-left corner  |
| <u>Top</u>                | Gets/Sets the Y coordinate of the upper-left corner  |
| <u>Right</u>              | Gets/Sets the X coordinate of the lower-right corner |
| <u>Bottom</u>             | Gets/Sets the Y coordinate of the lower-right corner |
| <u>Width</u>              | Gets/Sets the width                                  |
| <u>Height</u>             | Gets/Sets the height                                 |
| <u>SetBound</u>           | Sets new coordinates of using CRect class            |
| <u>Move</u>               | Sets new coordinates of the CRect class              |
| <u>Shift</u>              | Performs the relative shift of the CRect coordinates |
| <u>Contains</u>           | Checks if the point is inside the CRect class area   |
| <b>Additional methods</b> |                                                      |
| <u>Format</u>             | Gets the area coordinates as string                  |

## Left (Get Method)

Gets the X coordinate of the upper-left corner.

```
int Left()
```

### Returned value

X coordinate of the upper-left corner.

## Left (Set Method)

Sets the X coordinate of the upper-left corner.

```
void Left(  
    const int x    // new x coordinate  
)
```

### Parameters

x

[in] New X coordinate of the upper-left corner.

### Returned value

None.

## Top (Get Method)

Gets the Y coordinate of the upper-left corner.

```
int Top()
```

### Returned value

Y coordinate of the upper-left corner.

## Top (Set Method)

Sets the Y coordinate of the upper-left corner.

```
void Top(  
    const int y    // y coordinate  
)
```

### Parameters

*y*

[in] New Y coordinate of the upper-left corner.

### Returned value

None.

## Right (Get Method)

Gets the X coordinate of the lower-right corner.

```
int Right()
```

### Returned value

X coordinate of the lower-right corner.

## Right (Set Method)

Sets the Y coordinate of the lower-right corner.

```
void Right(  
    const int x    // x coordinate  
)
```

### Parameters

x

[in] New X coordinate of the lower-right corner.

### Returned value

None.

## Bottom (Get Method)

Gets the Y coordinate of the lower-right corner.

```
int Bottom()
```

### Returned value

Y coordinate of the lower-right corner.

## Bottom (Set Method)

Sets the Y coordinate of the lower-right corner.

```
void Bottom(  
    const int y    // y coordinate  
)
```

### Parameters

*y*

[in] New Y coordinate of the lower-right corner.

### Returned value

None.

## Width (Get Method)

Gets the width of the area.

```
int Width()
```

### Returned value

Width of the area.

## Width (Set Method)

Sets new width of the area.

```
virtual bool Width(  
    const int w    // width  
)
```

### Parameters

*w*

[in] New width.

### Returned value

true if successful, otherwise false.



## Height (Get Method)

Gets the height of the area.

```
int Height()
```

### Returned value

Height of the area.

## Height (Set Method)

Sets new height of the area.

```
virtual bool Height(  
    const int h    // height  
)
```

### Parameters

*h*

[in] New height.

### Returned value

true if successful, otherwise false.

## SetBound

Sets new coordinates of the area using CRect class coordinates.

```
void SetBound(  
    const & CRect rect    // CRect class  
)
```

### Returned value

None.

## SetBound

Sets new coordinates of the area.

```
void SetBound(  
    const int l    // left  
    const int t    // topt  
    const int r    // right  
    const int b    // bottom  
)
```

### Parameters

*l*

[in] X coordinate of the upper-left corner.

*t*

[in] Y coordinate of the upper-left corner.

*r*

[in] X coordinate of the lower-right corner.

*b*

[in] Y coordinate of the lower-right corner.

### Returned value

None.

## Move

Sets new coordinates of the CRect class.

```
void Move(  
    const int x,      // X coordinate  
    const int y      // Y coordinate  
)
```

### Parameters

*x*

[in] New X coordinate.

*y*

[in] New Y coordinate.

### Returned value

None.

## Shift

Performs the relative shift of the CRect class coordinates.

```
void Shift(  
    const int dx,      // delta X  
    const int dy      // delta Y  
)
```

### Parameters

*dx*

[in] Delta X.

*dy*

[in] Delta Y.

### Returned value

None.

## Contains

Checks if the point is inside the CRect class area.

```
bool Contains(  
    const int x,      // X coordinate  
    const int y      // Y coordinate  
)
```

### Parameters

*x*

[in] X coordinate.

*y*

[in] Y coordinate.

### Returned value

true, if the point is inside the area (including borders), otherwise - false.

## Format

Gets the area coordinates as string.

```
string Format(  
    string & fmt,      // format  
    ) const
```

### Parameters

*fmt*

[in] String with format.

### Returned value

String with the area coordinates.

## CWnd

CWnd is a base class for all controls, included in the MQL5 Standard Library.

### Description

CWnd class is the implementation of the base control class.

### Declaration

```
class CWnd : public CObject
```

### Title

```
#include <Controls\Wnd.mqh>
```

### Class Methods

| Create and destroy            |                                                                   |
|-------------------------------|-------------------------------------------------------------------|
| <a href="#">Create</a>        | Creates control                                                   |
| <a href="#">Destroy</a>       | Destroys control                                                  |
| Chart event handlers          |                                                                   |
| <a href="#">OnEvent</a>       | Event handler of all chart events                                 |
| <a href="#">OnMouseEvent</a>  | Event handler for the <a href="#">CHARTEVENT_MOUSE_MOVE</a> event |
| Name                          |                                                                   |
| <a href="#">Name</a>          | Gets control name                                                 |
| Access to container           |                                                                   |
| <a href="#">ControlsTotal</a> | Gets the number of controls in the container                      |
| <a href="#">Control</a>       | Gets the control by index                                         |
| <a href="#">ControlFind</a>   | Gets the control by ID                                            |
| Geometry                      |                                                                   |
| <a href="#">Rect</a>          | Gets the pointer to the CRect class object                        |
| <a href="#">Left</a>          | Gets/Sets the X-coordinate of the upper-left corner               |
| <a href="#">Top</a>           | Gets/Sets the Y-coordinate of the upper-left corner               |
| <a href="#">Right</a>         | Gets/Sets the X-coordinate of the lower-right corner              |
| <a href="#">Bottom</a>        | Gets/Sets the Y-coordinate of the lower-right corner              |

|                                 |                                                        |
|---------------------------------|--------------------------------------------------------|
| <a href="#">Width</a>           | Gets/Sets the width                                    |
| <a href="#">Height</a>          | Gets/Sets the height                                   |
| <a href="#">Move</a>            | Sets new coordinates of the control                    |
| <a href="#">Shift</a>           | Performs the relative shift of the control coordinates |
| <a href="#">Resize</a>          | Sets new width/height of the control                   |
| <a href="#">Contains</a>        | Checks if the point/control is inside the control area |
| <b>Align</b>                    |                                                        |
| <a href="#">Alignment</a>       | Sets alignment properties of the control               |
| <a href="#">Align</a>           | Performs control alignment                             |
| <b>Identification</b>           |                                                        |
| <a href="#">Id</a>              | Gets/Sets the control ID                               |
| <b>State</b>                    |                                                        |
| <a href="#">IsEnabled</a>       | Gets a value indicating whether the control is enabled |
| <a href="#">Enable</a>          | Sets a value indicating whether the control is enabled |
| <a href="#">Disable</a>         | Disables the control                                   |
| <a href="#">IsVisible</a>       | Checks the visibility flag                             |
| <a href="#">Visible</a>         | Sets the visibility flag                               |
| <a href="#">Show</a>            | Shows the control                                      |
| <a href="#">Hide</a>            | Hides the control                                      |
| <a href="#">IsActive</a>        | Checks the control activity                            |
| <a href="#">Activate</a>        | Activates the control                                  |
| <a href="#">Deactivate</a>      | Deactivates the control                                |
| <b>State flags</b>              |                                                        |
| <a href="#">StateFlags</a>      | Gets/Sets the control state flags                      |
| <a href="#">StateFlagsSet</a>   | Sets the control state flags                           |
| <a href="#">StateFlagsReset</a> | Resets the control state flags                         |
| <b>Properties flags</b>         |                                                        |
| <a href="#">PropFlags</a>       | Gets/Sets the control properties flags                 |
| <a href="#">PropFlagsSet</a>    | Sets the control properties flags                      |
| <a href="#">PropFlagsReset</a>  | Resets the control properties flags                    |



|                                          |                                    |
|------------------------------------------|------------------------------------|
| <b>Mouse operations</b>                  |                                    |
| <a href="#"><u>MouseX</u></a>            | Gets/Saves the mouse X coordinate  |
| <a href="#"><u>MouseY</u></a>            | Gets/Saves the mouse Y coordinate  |
| <a href="#"><u>MouseFlags</u></a>        | Gets/Saves the mouse buttons state |
| <a href="#"><u>MouseFocusKill</u></a>    | Kills mouse focus                  |
| <b>Internal event handlers</b>           |                                    |
| <a href="#"><u>OnCreate</u></a>          | "Create" event handler             |
| <a href="#"><u>OnDestroy</u></a>         | "Destroy" event handler            |
| <a href="#"><u>OnMove</u></a>            | "Move" event handler               |
| <a href="#"><u>OnResize</u></a>          | "Resize" event handler             |
| <a href="#"><u>OnEnable</u></a>          | "Enable" event handler             |
| <a href="#"><u>OnDisable</u></a>         | "Disable" event handler            |
| <a href="#"><u>OnShow</u></a>            | "Show" event handler               |
| <a href="#"><u>OnHide</u></a>            | "Hide" event handler               |
| <a href="#"><u>OnActivate</u></a>        | "Activate" event handler           |
| <a href="#"><u>OnDeactivate</u></a>      | "Deactivate" event handler         |
| <a href="#"><u>OnClick</u></a>           | "Click" event handler              |
| <a href="#"><u>OnChange</u></a>          | "Change" event handler             |
| <b>Mouse event handlers</b>              |                                    |
| <a href="#"><u>OnMouseDown</u></a>       | "MouseDown" event handler          |
| <a href="#"><u>OnMouseUp</u></a>         | "MouseUp" event handler            |
| <b>Drag event handlers</b>               |                                    |
| <a href="#"><u>OnDragStart</u></a>       | "DragStart" event handler          |
| <a href="#"><u>OnDragProcess</u></a>     | "DragProcess" event handler        |
| <a href="#"><u>OnDragEnd</u></a>         | "DragEnd" event handler            |
| <b>Drag object</b>                       |                                    |
| <a href="#"><u>DragObjectCreate</u></a>  | Creates drag object                |
| <a href="#"><u>DragObjectDestroy</u></a> | Destroys drag object               |

## Create

Creates a control.

```
virtual bool Create(  
    const long    chart,      // chart ID  
    const string  name,      // name  
    const int     subwin,     // chart subwindow  
    const int     x1,         // x1 coordinate  
    const int     y1,         // y1 coordinate  
    const int     x2,         // x2 coordinate  
    const int     y2         // y2 coordinate  
)
```

### Parameters

*chart*

[in] chart ID.

*name*

[in] Unique name of the control.

*subwin*

[in] Chart subwindow.

*x1*

[in] X coordinate of the upper-left corner.

*y1*

[in] Y coordinate of the upper-left corner.

*x2*

[in] X coordinate of the lower-right corner.

*y2*

[in] Y coordinate of the lower-right corner.

### Returned value

true if successful, otherwise false.

### Note

Base class method only saves the parameters and always return true.

## Destroy

Destroys a control.

```
virtual bool Destroy()
```

### Returned value

true if successful, otherwise false.

## OnEvent

Chart event handler.

```
virtual bool OnEvent(  
    const int      id,           // ID  
    const long&    lparam,       // event parameter of long type  
    const double&  dparam,       // event parameter of double type  
    const string&  sparam        // event parameter of string type  
)
```

### Parameters

*id*

[in] Event ID.

*lparam*

[in] Event parameter of [long](#) type, passed by reference.

*dparam*

[in] Event parameter of [double](#) type, passed by reference.

*sparam*

[in] Event parameter of [string](#) type, passed by reference.

### Returned value

true - if the event has been processed, otherwise false.

## OnMouseEvent

Mouse event handler (the [CHARTEVENT\\_MOUSE\\_MOVE](#) chart event).

```
virtual bool OnMouseEvent (  
    const int x,           // x coordinate  
    const int y,           // y coordinate  
    const int flags        // flags  
)
```

### Parameters

*x*

[in] X coordinate of the mouse cursor relative to the upper-left corner of the chart.

*y*

[in] Y coordinate of the mouse cursor relative to the upper-left corner of the chart.

*flags*

[in] Flag of mouse buttons states.

### Returned value

true - if the event has been processed, otherwise false.

## Name

Gets control name.

```
string Name() const
```

### Returned value

Control name.

## ControlsTotal

Gets the number of controls in the container.

```
int ControlsTotal() const
```

### Returned value

Number of controls in container.

### Note

The base class method doesn't have the container, it provides the access to container for its heirs and always returns 0.

## Control

Gets the control by index.

```
CWnd* Control(  
    const int ind    // index  
    ) const
```

### Parameters

*ind*

[in] Control index.

### Returned value

A pointer to the control.

### Note

The base class method doesn't have the container, it provides the access to container for its heirs and always returns NULL.



## ControlFind

Gets the control from container by specified ID.

```
virtual CWnd* ControlFind(  
    const long id    // ID  
)
```

### Parameters

*id*

[in] Identifier of the control to find.

### Returned value

Pointer to the control.

### Note

The base class method doesn't have the container, it provides the access to container for its heirs. If the specified ID match with the container ID, it returns a pointer to itself (this).

## Rect

Gets the pointer to the CRect class object.

```
const CRect* Rect () const
```

### Returned value

Pointer to the CRect class object.

## Left (Get Method)

Gets the X coordinate of the upper-left corner of the control.

```
int Left()
```

### Returned value

X coordinate of the upper-left corner of the control.

## Left (Set Method)

Sets the X coordinate of the upper-left corner of the control.

```
void Left(  
    const int x    // new x coordinate  
)
```

### Parameters

x

[in] New X coordinate of the upper-left corner.

### Returned value

None.

## Top (Get Method)

Gets the Y coordinate of the upper-left corner of the control.

```
int Top()
```

### Returned value

Y coordinate of the upper-left corner of the control.

## Top (Set Method)

Sets the Y coordinate of the upper-left corner of the control.

```
void Top(  
    const int y    // y coordinate  
)
```

### Parameters

*y*

[in] New Y coordinate of the upper-left corner.

### Returned value

None.

## Right (Get Method)

Gets the X coordinate of the lower-right corner of the control.

```
int Right()
```

### Returned value

X coordinate of the lower-right corner.

## Right (Set Method)

Sets the Y coordinate of the lower-right corner of the control.

```
void Right(  
    const int x    // x coordinate  
)
```

### Parameters

x

[in] New X coordinate of the lower-right corner.

### Returned value

None.

## Bottom (Get Method)

Gets the Y coordinate of the lower-right corner of the control.

```
int Bottom()
```

### Returned value

Y coordinate of the lower-right corner of the control.

## Bottom (Set Method)

Sets the Y coordinate of the lower-right corner of the control.

```
void Bottom(  
    const int y    // y coordinate  
)
```

### Parameters

*y*

[in] New Y coordinate of the lower-right corner.

### Returned value

None.

## Width (Get Method)

Gets the control width.

```
int Width()
```

### Returned value

Width of the control.

## Width (Set Method)

Sets new width of the control.

```
virtual bool Width(  
    const int w    // width  
)
```

### Parameters

*w*

[in] New width.

### Returned value

true if successful, otherwise false.

## Height (Get Method)

Gets the control height.

```
int Height()
```

### Returned value

Height of the control.

## Height (Set Method)

Sets new height of the control.

```
virtual bool Height(  
    const int h    // height  
)
```

### Parameters

*h*

[in] New height.

### Returned value

true if successful, otherwise false.



## Move

Sets new coordinates of the control.

```
void Move(  
    const int x,      // X coordinate  
    const int y      // Y coordinate  
)
```

### Parameters

*x*

[in] New X coordinate.

*y*

[in] New Y coordinate.

### Returned value

None.

## Shift

Performs the relative shift of the control coordinates.

```
void Shift(  
    const int dx,      // delta X  
    const int dy      // delta Y  
)
```

### Parameters

*dx*

[in] Delta X.

*dy*

[in] Delta Y.

### Returned value

None.

## Resize

Sets new width/height of the control.

```
virtual bool  Resize(  
    const int  w,      // width  
    const int  h      // height  
)
```

### Parameters

*w*

[in] New width.

*h*

[in] New height.

### Returned value

true if successful, otherwise false.

## Contains

Checks if the point is inside the control area of the chart.

```
bool Contains(  
    const int x,      // X coordinate  
    const int y      // Y coordinate  
)
```

### Parameters

*x*

[in] X coordinate.

*y*

[in] Y coordinate.

### Returned value

true, if the point is inside the area (including borders), otherwise - false.

## Contains

Checks if the specified control is inside the control area of the chart.

```
bool Contains(  
    const CWnd* control    // pointer  
) const
```

### Parameters

*control*

[in] Object pointer.

### Returned value

true, if the specified control is inside the area (including borders), otherwise - false.

## Alignment

Sets alignment parameters of the control.

```
void Alignment(  
    const int  flags,      // alignment flags  
    const int  left,       // offset from the left border  
    const int  top,        // offset from the top border  
    const int  right,      // offset from the right border  
    const int  bottom      // offset from the bottom border  
)
```

### Parameters

*flags*

[in] Alignment flags.

*left*

[in] Fixed offset from the left border.

*top*

[in] Fixed offset from the top border.

*right*

[in] Fixed offset from the right border.

*bottom*

[in] Fixed offset from the bottom border.

### Returned value

None.

### Note

Alignement flags:

```
enum WND_ALIGN_FLAGS  
{  
    WND_ALIGN_NONE=0,           // no align  
    WND_ALIGN_LEFT=1,          // align left  
    WND_ALIGN_TOP=2,           // align top  
    WND_ALIGN_RIGHT=4,         // align right  
    WND_ALIGN_BOTTOM=8,        // align bottom  
    WND_ALIGN_WIDTH = WND_ALIGN_LEFT|WND_ALIGN_RIGHT, // align width  
    WND_ALIGN_HEIGHT=WND_ALIGN_TOP|WND_ALIGN_BOTTOM,  // align height  
    WND_ALIGN_CLIENT=WND_ALIGN_WIDTH|WND_ALIGN_HEIGHT, // align height and width  
}
```

## Align

Performs control alignment in the specified chart area.

```
virtual bool Align(  
    const CRect* rect    // pointer  
)
```

### Parameters

*rect*

[in] Pointer to the object with chart area coordinates.

### Returned value

true if successful, otherwise false.

### Note

The alignment parameters must be specified (no alignment by default).

## Id (Get Method)

Gets the control ID.

```
long Id() const
```

### Returned value

The control identifier.

## Id (Set Method)

Sets new value of the control ID.

```
virtual long Id(  
    const long id    // identifier  
)
```

### Parameters

*id*

[in] New value of the control identifier.

### Returned value

None.

## IsEnabled

Gets a value indicating whether the control is enabled.

```
bool  IsEnabled()  const
```

### Returned value

true - if the control is enabled, otherwise - false.



## Enable

Enables the control.

```
virtual bool Enable()
```

### Returned value

true if successful, otherwise false.

### Note

If the control is enabled, it's able to process the external events.

## Disable

Disables the control.

```
virtual bool Disable()
```

### Returned value

true if successful, otherwise false.

### Note

The disabled control doesn't able to process the external events.

## IsVisible

Gets a value indicating whether the control is visible.

```
bool IsVisible() const
```

### Returned value

true if the control is shown on the chart, otherwise false.

## Visible

Sets the visibility flag.

```
virtual bool Visible(  
    const bool flag    // flag  
)
```

### Parameters

*flag*

[in] New flag.

### Returned value

true if successful, otherwise false.

## Show

Shows the control.

```
virtual bool Show()
```

### Returned value

true if successful, otherwise false.

## Hide

Hides the control.

```
virtual bool Hide()
```

### Returned value

true if successful, otherwise false.

## IsActive

Gets a value indicating whether the control is active.

```
bool IsActive() const
```

### Returned value

true if the control is active, otherwise false.

## Activate

Activates the control.

```
virtual bool Activate()
```

### Returned value

true if successful, otherwise false.

### Note

The control becomes active when the mouse cursor is hovering over it.



## Deactivate

Deactivates the control.

```
virtual bool Deactivate()
```

### Returned value

true if successful, otherwise false.

### Note

The control becomes inactive when the mouse cursor is out off the control.

## StateFlags (Get Method)

Gets the control state flags.

```
int StateFlags()
```

### Returned value

The control state flags.

## StateFlags (Set Method)

Sets the control state flags.

```
virtual void StateFlags (  
    const int flags    // flags  
)
```

### Parameters

*flags*

[in] New control state flags.

### Returned value

None.

## StateFlagsSet

Sets the control state flags.

```
virtual void StateFlagsSet(  
    const int flags    // flags  
)
```

### Parameters

*flags*

[in] Flags to set (bit mask).

### Returned value

None.

## StateFlagsReset

Resets the control state flags.

```
virtual void StateFlagsReset(  
    const int flags    // flags  
)
```

### Parameters

*flags*

[in] Flags to reset (bit mask).

### Returned value

None.

## PropFlags (Get Method)

Gets the control properties flags.

```
void PropFlags(  
    const int flags    // flags  
)
```

### Returned value

The control properties flags.

## PropFlags (Set Method)

Sets the control properties flags.

```
virtual void PropFlags(  
    const int flags    // flags  
)
```

### Parameters

*flags*

[in] New flags.

### Returned value

None.

## PropFlagsSet

Sets the control properties flags.

```
virtual void PropFlagsSet (  
    const int flags    // flags  
)
```

### Parameters

*flags*

[in] Flags to set (bit mask).

### Returned value

None.

## PropFlagsReset

Resets the control properties flags.

```
virtual void PropFlagsReset(  
    const int flags    // flags  
)
```

### Parameters

*flags*

[in] Flags to reset (bit mask).

### Returned value

None.

## MouseX (Set Method)

Saves the mouse X coordinate.

```
void MouseX(  
    const int value    // coordinate  
)
```

### Parameters

*value*

[in] The X coordinate of the mouse.

### Returned value

None.

## MouseX (Get Method)

Gets the saved X coordinate of the mouse.

```
int MouseX()
```

### Returned value

The saved X coordinate of the mouse.



## MouseY (Set Method)

Saves the mouse Y coordinate.

```
void MouseY(  
    const int value    // coordinate  
)
```

### Parameters

*value*

[in] The Y coordinate of the mouse.

### Returned value

None.

## MouseY (Get Method)

Gets the saved Y coordinate of the mouse.

```
int MouseY()
```

### Returned value

The saved Y coordinate of the mouse.

## MouseFlags (Set Method)

Saves the state of mouse buttons.

```
virtual void MouseFlags(  
    const int value    // state  
)
```

### Parameters

*value*

[in] State of mouse buttons.

### Returned value

None.

## MouseFlags (Get Method)

Gets the saved state of mouse buttons.

```
int MouseFlags()
```

### Returned value

State of mouse buttons.

## MouseFocusKill

Clears the saved state of mouse buttons and deactivates the control.

```
bool MouseFocusKill(  
    const long id=CONTROLS_INVALID_ID    // id  
)
```

### Parameters

*id=CONTROLS\_INVALID\_ID*

[in] Identifier of the control, that received mouse focus.

### Returned value

The control deactivation result.

## OnCreate

The control "Create" event handler.

```
virtual bool OnCreate()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The base class method does nothing and always returns true.

## OnDestroy

The control "Destroy" event handler.

```
virtual bool OnDestroy()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The base class method does nothing and always returns true.

## OnMove

The control "Move" event handler.

```
virtual bool OnMove()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The base class method does nothing and always returns true.

## OnResize

The control "Resize" event handler.

```
virtual bool OnResize()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The base class method does nothing and always returns true.

## OnEnable

The control "Enable" (if enabled, it can respond to user interaction) event handler.

```
virtual bool OnEnable()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The base class method does nothing and always returns true.



## OnDisable

The control "Disable" (if disabled it can't respond to user interaction) event handler.

```
virtual bool OnDisable()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The base class method does nothing and always returns true.

## OnShow

The control "Show" event handler.

```
virtual bool OnShow()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The base class method does nothing and always returns true.

## OnHide

The control "Hide" event handler.

```
virtual bool OnHide()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The base class method does nothing and always returns true.

## OnActivate

The control "Activate" event handler.

```
virtual bool OnActivate()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The base class method does nothing and always returns true.

## OnDeactivate

The control "Deactivate" event handler.

```
virtual bool OnDeactivate()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The base class method does nothing and always returns true.

## OnClick

The control "Click" (left mouse button click) event handler.

```
virtual bool OnClick()
```

### Returned value

true if the event has been processed, otherwise false.

## OnChange

The control "Change" event handler.

```
virtual bool OnChange()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The base class method does nothing and always returns true.

## OnMouseDown

The control "MouseDown" event handler.

```
virtual bool OnMouseDown()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The "MouseDown" event occurs when left mouse button is pressed on the control.



## OnMouseUp

The control "MouseUp" (left mouse button release) event handler.

```
virtual bool OnMouseUp()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The "MouseUp" event occurs when left mouse button is released on the control.

## OnDragStart

The control "DragStart" event handler.

```
virtual bool OnDragStart()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The "DragStart" event occurs at the start of drag operation.

## OnDragProcess

The control "DragProcess" event handler.

```
virtual bool OnDragProcess(  
    const int x,      // x coordinate  
    const int y      // y coordinate  
)
```

### Parameters

*x*

[in] Current X coordinate of mouse cursor.

*y*

[in] Current Y coordinate of mouse cursor.

### Returned value

true if the event has been processed, otherwise false.

### Note

The "DragProcess" event occurs when the control is moved.

## OnDragEnd

The control "DragEnd" event handler.

```
virtual bool OnDragEnd()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The "DragEnd" event occurs when control drag process is finished.

## DragObjectCreate

Creates drag object.

```
virtual bool DragObjectCreate ()
```

### Returned value

true if successful, otherwise false.

### Note

true if the event has been processed, otherwise false.

## DragObjectDestroy

Destroys drag object.

```
virtual bool DragObjectDestroy()
```

### Returned value

true if successful, otherwise false.

## CWndContainer

CWndContainer is a base class for a complex control (containing dependent controls) of the Standard library.

### Description

CWndContainer class implements base methods of the complex control.

### Declaration

```
class CWndContainer : public CWnd
```

### Title

```
#include <Controls\WndContainer.mqh>
```

### Class Methods

| Destroy                       |                                                                                  |
|-------------------------------|----------------------------------------------------------------------------------|
| <a href="#">Destroy</a>       | Destroys all the container controls                                              |
| Chart event handlers          |                                                                                  |
| <a href="#">OnEvent</a>       | Event handler of all chart events                                                |
| <a href="#">OnMouseEvent</a>  | The <a href="#">CHARTEVENT_MOUSE_MOVE</a> event handler                          |
| Access to container           |                                                                                  |
| <a href="#">ControlsTotal</a> | Gets the number controls in the container                                        |
| <a href="#">Control</a>       | Gets control by index                                                            |
| <a href="#">ControlFind</a>   | Gets control by ID                                                               |
| Add/Delete                    |                                                                                  |
| <a href="#">Add</a>           | Adds control to container                                                        |
| <a href="#">Delete</a>        | Deletes control from container                                                   |
| Geometry                      |                                                                                  |
| <a href="#">Move</a>          | Sets new coordinates for all controls of the container                           |
| <a href="#">Shift</a>         | Performs the relative shift of the coordinates for all controls of the container |
| Identification                |                                                                                  |
| <a href="#">Id</a>            | Sets the ID for all controls of the container                                    |
| State                         |                                                                                  |
| <a href="#">Enable</a>        | Enables all controls of the container                                            |

|                                |                                        |
|--------------------------------|----------------------------------------|
| <a href="#">Disable</a>        | Disables all controls of the container |
| <a href="#">Show</a>           | Shows all controls of the container    |
| <a href="#">Hide</a>           | Hides all controls of the container    |
| <b>Mouse operations</b>        |                                        |
| <a href="#">MouseFocusKill</a> | Kills mouse focus                      |
| <b>File operations</b>         |                                        |
| <a href="#">Save</a>           | Saves container information to file    |
| <a href="#">Load</a>           | Loads container information from file  |
| <b>Internal event handlers</b> |                                        |
| <a href="#">OnResize</a>       | "Resize" event handler                 |
| <a href="#">OnActivate</a>     | "Activate" event handler               |
| <a href="#">OnDeactivate</a>   | "Deactivate" event handler             |



## Destroy

Destroys all the container controls.

```
virtual bool Destroy()
```

### Returned value

true if successful, otherwise false.

## OnEvent

Chart event handler.

```
virtual bool OnEvent(  
    const int      id,           // ID  
    const long&    lparam,      // event parameter of long type  
    const double&  dparam,      // event parameter of double type  
    const string&  sparam       // event parameter of string type  
)
```

### Parameters

*id*

[in] Event ID.

*lparam*

[in] Event parameter of [long](#) type, passed by reference.

*dparam*

[in] Event parameter of [double](#) type, passed by reference.

*sparam*

[in] Event parameter of [string](#) type, passed by reference.

### Returned value

true - if the event has been processed, otherwise false.

## OnMouseEvent

Mouse event handler.

```
virtual bool OnMouseEvent (  
    const int x,           // x coordinate  
    const int y,           // y coordinate  
    const int flags        // flags  
)
```

### Parameters

*x*

[in] X coordinate of the mouse cursor relative to the upper-left corner of the chart.

*y*

[in] Y coordinate of the mouse cursor relative to the upper-left corner of the chart.

*flags*

[in] Flag of mouse buttons states.

### Returned value

true - if the event has been processed, otherwise false.

## ControlsTotal

Gets the number of controls in the container.

```
int ControlsTotal() const
```

### Returned value

Number of controls in the container.

## Control

Gets control from the container by index.

```
CWnd* Control(  
    const int ind    // index  
) const
```

### Parameters

*ind*

[in] Index of the control needed.

### Returned value

Pointer to the control, otherwise NULL if the control isn't found.

## ControlFind

Gets control from the container by identifier.

```
virtual CWnd* ControlFind(  
    const long id    // id  
)
```

### Parameters

*id*

[in] Control ID.

### Returned value

Pointer to the control, otherwise NULL if the control isn't found.

## Add

Adds a control to the container.

```
bool Add(  
    CWnd& control    // reference  
)
```

### Parameters

*control*

[in] Control to add, passed by reference.

### Returned value

true if successful, otherwise false.

## Delete

Deletes control from the container.

```
bool Delete(  
    CWnd& control    // reference  
)
```

### Parameters

*control*

[in] Control to delete, passed by reference.

### Returned value

true if successful, otherwise false.



## Move

Sets new coordinates for all controls of the container.

```
virtual bool Move(  
    const int x,      // x coordinate  
    const int y      // y coordinate  
)
```

### Parameters

*x*

[in] New X coordinate of the upper-left corner.

*y*

[in] New Y coordinate of the upper-left corner.

### Returned value

true if successful, otherwise false.

## Shift

Performs the relative shift of the coordinates for all controls of the container.

```
virtual bool Shift(  
    const int dx,      // delta x  
    const int dy      // delta y  
)
```

### Parameters

*dx*

[in] Delta X.

*dy*

[in] Delta Y.

### Returned value

true if successful, otherwise false.

## Id

Sets the ID for all controls of the container.

```
virtual long Id(  
    const long id    // identifier  
)
```

### Parameters

*id*

[in] Base group identifier.

### Returned value

Number of identifiers, used by container controls.

## Enable

Enables all the controls of the container.

```
virtual bool Enable()
```

### Returned value

true if successful, otherwise false.

## Disable

Disables all controls of the container.

```
virtual bool Disable()
```

### Returned value

true if successful, otherwise false.

## Show

Shows all controls of the container.

```
virtual bool Show()
```

### Returned value

true if successful, otherwise false.

## Hide

Hides all controls of the container.

```
virtual bool Hide()
```

### Returned value

true if successful, otherwise false.

## MouseFocusKill

Clears the saved state of mouse buttons and deactivates all controls in the container.

```
bool MouseFocusKill(  
    const long id=CONTROLS_INVALID_ID    // id  
)
```

### Parameters

*id=CONTROLS\_INVALID\_ID*

[in] Identifier of the control, that received mouse focus.

### Returned value

The controls deactivation result.



## Save

Saves container information to file.

```
virtual bool Save(  
    const int file_handle    // handle  
)
```

### Parameters

*file\_handle*

[in] Handle of the binary file (must be opened for writing).

### Returned value

true if successful, otherwise false.

## Load

Loads container information to file

```
virtual bool Load(  
    const int file_handle    // handle  
)
```

### Parameters

*file\_handle*

[in] Handle of the binary file (must be opened for reading).

### Returned value

true if successful, otherwise false.

## OnResize

The control "Resize" event handler.

```
virtual bool OnResize()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The base class method does nothing and always returns true.

## OnActivate

The control "Activate" event handler.

```
virtual bool OnActivate()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The base class method does nothing and always returns true.

## OnDeactivate

The control "Deactivate" event handler.

```
virtual bool OnDeactivate()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The base class method does nothing and always returns true.

## CWndObj

CWndObj is a base class for simple controls (based on chart objects) of the Standard library.

### Description

CWndObj class implements base methods of the simple control.

### Declaration

```
class CWndObj : public CWnd
```

### Title

```
#include <Controls\WndObj.mqh>
```

### Class Methods

| Chart events processing          |                                                                                 |
|----------------------------------|---------------------------------------------------------------------------------|
| <a href="#">OnEvent</a>          | Event handler of all chart events                                               |
| Properties                       |                                                                                 |
| <a href="#">Text</a>             | Gets/Sets the <a href="#">OBJPROP_TEXT</a> property of the chart object         |
| <a href="#">Color</a>            | Gets/Sets the <a href="#">OBJPROP_COLOR</a> property of the chart object        |
| <a href="#">ColorBackground</a>  | Gets/Sets the <a href="#">OBJPROP_BGCOLOR</a> property of the chart object      |
| <a href="#">ColorBorder</a>      | Gets/Sets the <a href="#">OBJPROP_BORDER_COLOR</a> property of the chart object |
| <a href="#">Font</a>             | Gets/Sets the <a href="#">OBJPROP_FONT</a> property of the chart object         |
| <a href="#">FontSize</a>         | Gets/Sets the <a href="#">OBJPROP_FONTSIZE</a> property of the chart object     |
| <a href="#">ZOrder</a>           | Gets/Sets the <a href="#">OBJPROP_ZORDER</a> property of the chart object       |
| Chart objects event handlers     |                                                                                 |
| <a href="#">OnObjectCreate</a>   | <a href="#">CHARTEVENT_OBJECT_CREATE</a> event handler                          |
| <a href="#">OnObjectChange</a>   | <a href="#">CHARTEVENT_OBJECT_CHANGE</a> event handler                          |
| <a href="#">OnObjectDelete</a>   | <a href="#">CHARTEVENT_OBJECT_DELETE</a> event handler                          |
| <a href="#">OnObjectDrag</a>     | <a href="#">CHARTEVENT_OBJECT_DRAG</a> event handler                            |
| Properties change event handlers |                                                                                 |
| <a href="#">OnSetText</a>        | "SetText" event handler                                                         |

|                                             |                                    |
|---------------------------------------------|------------------------------------|
| <a href="#"><u>OnSetColor</u></a>           | "SetColor" event handler           |
| <a href="#"><u>OnSetColorBackground</u></a> | "SetColorBackground" event handler |
| <a href="#"><u>OnSetFont</u></a>            | "SetFont" event handler            |
| <a href="#"><u>OnSetFontSize</u></a>        | "SetFontSize" event handler        |
| <a href="#"><u>OnSetZOrder</u></a>          | "SetZOrder" event handler          |
| Internal event handlers                     |                                    |
| <a href="#"><u>OnDestroy</u></a>            | "Destroy" event handler            |
| <a href="#"><u>OnChange</u></a>             | "Change" event handler             |

## OnEvent

Chart event handler.

```
virtual bool OnEvent(  
    const int      id,           // ID  
    const long&    lparam,       // event parameter of long type  
    const double&  dparam,       // event parameter of double type  
    const string&  sparam        // event parameter of string type  
)
```

### Parameters

*id*

[in] Event ID.

*lparam*

[in] Event parameter of [long](#) type, passed by reference.

*dparam*

[in] Event parameter of [double](#) type, passed by reference.

*sparam*

[in] Event parameter of [string](#) type, passed by reference.

### Returned value

true - if the event has been processed, otherwise false.



## Text (Get method)

Gets the [OBJPROP\\_TEXT](#) (text) property of the chart object.

```
string Text()
```

### Returned value

The value of the [OBJPROP\\_TEXT](#) property.

## Text (Set method)

Sets the [OBJPROP\\_TEXT](#) (text) property of the chart object.

```
bool Text (  
    const string value    // new value  
)
```

### Parameters

*value*

[in] New value of the [OBJPROP\\_TEXT](#) property.

### Returned value

true if successful, otherwise false.

## Color (Get method)

Gets the [OBJPROP\\_COLOR](#) (color) property of the chart object.

```
color Color()
```

### Returned value

The value of the [OBJPROP\\_COLOR](#) property.

## Color (Set method)

Sets the [OBJPROP\\_COLOR](#) (color) property of the chart object.

```
bool Color(  
    const color value    // value  
)
```

### Parameters

*value*

[in] New value of the [OBJPROP\\_COLOR](#) property.

### Returned value

true if successful, otherwise false.

## ColorBackground (Get method)

Gets the [OBJPROP\\_BGCOLOR](#) (background color) of the chart object.

```
color ColorBackground()
```

### Returned value

The value of the [OBJPROP\\_BGCOLOR](#) property.

## ColorBackground (Set method)

Sets the [OBJPROP\\_BGCOLOR](#) (background color) property of the chart object.

```
bool ColorBackground(  
    const color value    // value  
)
```

### Parameters

*value*

[in] New value of the [OBJPROP\\_BGCOLOR](#) property.

### Returned value

true if successful, otherwise false.

## ColorBorder (Get method)

Gets the [OBJPROP\\_BORDER\\_COLOR](#) (border color) property of the chart object.

```
color ColorBorder ()
```

### Returned value

The value of the [OBJPROP\\_BORDER\\_COLOR](#) property.

## ColorBorder (Set method)

Sets the [OBJPROP\\_BORDER\\_COLOR](#) (border color) property of the chart object.

```
bool ColorBorder (  
    const color value    // value  
)
```

### Parameters

*value*

[in] New value of the [OBJPROP\\_BORDER\\_COLOR](#) property.

### Returned value

true if successful, otherwise false.

## Font (Get method)

Gets the [OBJPROP\\_FONT](#) (font) property of the chart object.

```
string Font()
```

### Returned value

The value of the [OBJPROP\\_FONT](#) property.

## Font (Set method)

Sets the [OBJPROP\\_FONT](#) (font) property of the chart object.

```
bool Font(  
    const string value    // new value  
)
```

### Parameters

*value*

[in] New value of the [OBJPROP\\_FONT](#) property.

### Returned value

true if successful, otherwise false.

## FontSize (Get method)

Gets the [OBJPROP\\_FONTSIZE](#) (font size) property of the chart object.

```
int FontSize()
```

### Returned value

The value of the [OBJPROP\\_FONTSIZE](#) property.

## FontSize (Set method)

Sets the [OBJPROP\\_FONTSIZE](#) (font size) property of the chart object.

```
bool FontSize(  
    const int value    // new value  
)
```

### Parameters

*value*

[in] New value of the [OBJPROP\\_FONTSIZE](#) property.

### Returned value

true if successful, otherwise false.

## ZOrder (Get method)

Gets the [OBJPROP\\_ZORDER](#) property of the chart object.

```
long ZOrder()
```

### Returned value

The value of the [OBJPROP\\_ZORDER](#) property.

## ZOrder (Set method)

Sets the [OBJPROP\\_ZORDER](#) property of the chart object.

```
bool ZOrder(  
    const long value // new value  
)
```

### Parameters

*value*

[in] New value of the [OBJPROP\\_ZORDER](#) property.

### Returned value

true if successful, otherwise false.

## OnObjectCreate

The chart object [CHARTEVENT\\_OBJECT\\_CREATE](#) event handler.

```
virtual bool OnObjectCreate()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The base class method does nothing and always returns true.



## OnObjectChange

The chart object [CHARTEVENT\\_OBJECT\\_CHANGE](#) event handler.

```
virtual bool OnObjectChange ()
```

### Returned value

true if the event has been processed, otherwise false.

## OnObjectDelete

The chart object [CHARTEVENT\\_OBJECT\\_DELETE](#) event handler.

```
virtual bool OnObjectDelete()
```

### Returned value

true if the event has been processed, otherwise false.

## OnObjectDrag

The chart object [CHARTEVENT\\_OBJECT\\_DRAG](#) event handler.

```
virtual bool OnObjectDrag()
```

### Returned value

true if the event has been processed, otherwise false.

## OnSetText

The control "SetText" (change of the [OBJPROP\\_TEXT](#) property) event handler.

```
virtual bool OnSetText()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The base class method does nothing and always returns true.

## OnSetColor

The control "SetColor" (change of the [OBJPROP\\_COLOR](#) property) event handler.

```
virtual bool OnSetColor()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The base class method does nothing and always returns true.

## OnSetColorBackground

The control "SetColorBackground" (change of the [OBJPROP\\_BGCOLOR](#) property) event handler.

```
virtual bool OnSetColorBackground()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The base class method does nothing and always returns true.

## OnSetFont

The control "SetFont" (change of the [OBJPROP\\_FONT](#) property) event handler.

```
virtual bool OnSetFont()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The base class method does nothing and always returns true.

## OnSetFontSize

The control "SetFontSize" (change of the [OBJPROP\\_FONTSIZE](#) property) event handler.

```
virtual bool OnSetFontSize()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The base class method does nothing and always returns true.



## OnSetZOrder

The control "SetZOrder" (change of the [OBJPROP\\_ZORDER](#) property) event handler.

```
virtual bool OnSetZOrder()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The base class method does nothing and always returns true.

## OnDestroy

The control "Destroy" event handler.

```
virtual bool OnDestroy()
```

### Returned value

true if the event has been processed, otherwise false.

## OnChange

The control "Change" event handler.

```
virtual bool OnChange()
```

### Returned value

true if the event has been processed, otherwise false.

## CLabel

CLabel is class of the simple control, based on "Text label" chart object.

### Description

CLabel is intended for creation of simple text labels.

### Declaration

```
class CLabel : public CWndObj
```

### Title

```
#include <Controls\Label.mqh>
```

### Class Methods

| Create                           |                             |
|----------------------------------|-----------------------------|
| <a href="#">Create</a>           | Creates control             |
| Properties change event handlers |                             |
| <a href="#">OnSetText</a>        | "SetText" event handler     |
| <a href="#">OnSetColor</a>       | "SetColor" event handler    |
| <a href="#">OnSetFont</a>        | "SetFont" event handler     |
| <a href="#">OnSetFontSize</a>    | "SetFontSize" event handler |
| Internal event handlers          |                             |
| <a href="#">OnCreate</a>         | "Create" event handler      |
| <a href="#">OnShow</a>           | "Show" event handler        |
| <a href="#">OnHide</a>           | "Hide" event handler        |
| <a href="#">OnMove</a>           | "Move" event handler        |

## Create

Creates new CLabel control.

```
virtual bool Create(  
    const long    chart,      // chart ID  
    const string  name,      // name  
    const int     subwin,     // chart subwindow  
    const int     x1,        // x1 coordinate  
    const int     y1,        // y1 coordinate  
    const int     x2,        // x2 coordinate  
    const int     y2         // y2 coordinate  
)
```

### Parameters

*chart*

[in] chart ID.

*name*

[in] Unique name of the control.

*subwin*

[in] Chart subwindow.

*x1*

[in] X coordinate of the upper left corner.

*y1*

[in] Y coordinate of the upper left corner.

*x2*

[in] X coordinate of the lower right corner.

*y2*

[in] Y coordinate of the lower right corner.

### Returned value

true if successful, otherwise false.

## OnSetText

The control "SetText" (change of the [OBJPROP\\_TEXT](#) property) event handler.

```
virtual bool OnSetText()
```

### Returned value

true if the event has been processed, otherwise false.

## OnSetColor

The control "SetColor" (change of the [OBJPROP\\_COLOR](#) property) event handler.

```
virtual bool OnSetColor()
```

### Returned value

true if the event has been processed, otherwise false.

## OnSetFont

The control "SetFont" (change of the [OBJPROP\\_FONT](#) property) event handler.

```
virtual bool OnSetFont()
```

### Returned value

true if the event has been processed, otherwise false.



## OnSetFontSize

The control "SetFontSize" (change of the [OBJPROP\\_FONTSIZE](#) property) event handler.

```
virtual bool OnSetFontSize()
```

### Returned value

true if the event has been processed, otherwise false.

## OnCreate

The control "Create" event handler.

```
virtual bool OnCreate()
```

### Returned value

true if the event has been processed, otherwise false.

## OnShow

The control "Show" event handler.

```
virtual bool OnShow()
```

### Returned value

true if the event has been processed, otherwise false.

## OnHide

The control "Hide" event handler.

```
virtual bool OnHide()
```

### Returned value

true if the event has been processed, otherwise false.

## OnMove

The control "Move" event handler.

```
virtual bool OnMove()
```

### Returned value

true if the event has been processed, otherwise false.

## CBmpButton

CBmpButton is class of the simple control, based on "Bitmap label" chart object.

### Description

CBmpButton is intended for creation of buttons with graphic image.

### Declaration

```
class CBmpButton : public CWndObj
```

### Title

```
#include <Controls\BmpButton.mqh>
```

### Class Methods

| Create                         |                                                      |
|--------------------------------|------------------------------------------------------|
| <a href="#">Create</a>         | Creates control                                      |
| Properties                     |                                                      |
| <a href="#">Border</a>         | Gets/Sets the "Border" property of the control       |
| <a href="#">BmpNames</a>       | Sets the name of bmp files of the control            |
| <a href="#">BmpOffName</a>     | Gets/Sets the name of bmp file for the OFF state     |
| <a href="#">BmpOnName</a>      | Gets/Sets the name of bmp file for the ON state      |
| <a href="#">BmpPassiveName</a> | Gets/Sets the name of bmp file for the passive state |
| <a href="#">BmpActiveName</a>  | Gets/Sets the name of bmp file for the active state  |
| State                          |                                                      |
| <a href="#">Pressed</a>        | Gets/Sets the state of the control                   |
| <a href="#">Locking</a>        | Gets/Sets the "Locking" property of the control      |
| Internal event handlers        |                                                      |
| <a href="#">OnSetZOrder</a>    | "SetZOrder" event handler                            |
| <a href="#">OnCreate</a>       | "Create" event handler                               |
| <a href="#">OnShow</a>         | "Show" event handler                                 |
| <a href="#">OnHide</a>         | "Hide" event handler                                 |
| <a href="#">OnMove</a>         | "Move" event handler                                 |

|                                     |                            |
|-------------------------------------|----------------------------|
| <a href="#"><u>OnChange</u></a>     | "Change" event handler     |
| <a href="#"><u>OnActivate</u></a>   | "Activate" event handler   |
| <a href="#"><u>OnDeactivate</u></a> | "Deactivate" event handler |
| <a href="#"><u>OnMouseDown</u></a>  | "MouseDown" event handler  |
| <a href="#"><u>OnMouseUp</u></a>    | "MouseUp" event handler    |

## Create

Creates new CBmpButton control.

```
virtual bool Create(  
    const long    chart,      // chart ID  
    const string  name,      // name  
    const int     subwin,     // chart subwindow  
    const int     x1,        // x1 coordinate  
    const int     y1,        // y1 coordinate  
    const int     x2,        // x2 coordinate  
    const int     y2         // y2 coordinate  
)
```

### Parameters

*chart*

[in] chart ID.

*name*

[in] Unique name of the control.

*subwin*

[in] Chart subwindow.

*x1*

[in] X coordinate of the upper left corner.

*y1*

[in] Y coordinate of the upper left corner.

*x2*

[in] X coordinate of the lower right corner.

*y2*

[in] Y coordinate of the lower right corner.

### Returned value

true if successful, otherwise false.



## Border (Get method)

Gets the "Border" (border width) property of the control.

```
int Border() const
```

### Returned value

The "Border" property.

## Border (Set method)

Sets the "Border" (border width) property of the control.

```
bool Border(  
    const int value    // new value  
)
```

### Parameters

*value*

[in] New value of the "Border" property.

### Returned value

true if successful, otherwise false.

## BmpNames

Sets the name of bmp files of the control

```
bool BmpNames(  
    const string off="",      // file name  
    const string on=""        // file name  
)
```

### Parameters

*off=""*

[in] Name of bmp file for OFF state.

*on=""*

[in] Name of bmp file for ON state.

### Returned value

true if successful, otherwise false.

## BmpOffName (Get method)

Gets the name of bmp file for OFF state.

```
string BmpOffName() const
```

### Returned value

Name of bmp file for OFF state.

## BmpOffName (Set method)

Sets the name of bmp file for OFF state.

```
bool BmpOffName (
    const string name    // file name
)
```

### Parameters

*name*

[in] Name of bmp file for OFF state.

### Returned value

true if successful, otherwise false.

## BmpOnName (Get method)

Gets the name of bmp file for ON state.

```
string BmpOnName() const
```

### Returned value

Name of bmp file for ON state.

## BmpOnName (Set method)

Sets the name of bmp file for ON state.

```
bool BmpOnName(  
    const string name    // file name  
)
```

### Parameters

*name*

[in] Name of bmp file for ON state.

### Returned value

true if successful, otherwise false.

## BmpPassiveName (Get method)

Gets the name of bmp file for the control passive state.

```
string BmpPassiveName() const
```

### Returned value

Name of bmp file for the control passive state.

## BmpPassiveName (Set method)

Sets the name of bmp file for the passive state.

```
bool BmpPassiveName(  
    const string name    // file name  
)
```

### Parameters

*name*

[in] Name of bmp file for the control passive state.

### Returned value

true if successful, otherwise false.

## BmpActiveName (Get method)

Gets the name of bmp file for the active state.

```
string BmpActiveName() const
```

### Returned value

Name of bmp file for the active state.

### Note

The control becomes active when the mouse cursor is hovering over it.

## BmpActiveName (Set method)

Sets the name of bmp file for the active state.

```
bool BmpActiveName(  
    const string name    // file name  
)
```

### Parameters

*name*

[in] Name of bmp file for the active state.

### Returned value

true if successful, otherwise false.

## Pressed (Get method)

Gets the state ("Pressed" property) of the control.

```
bool Pressed() const
```

### Returned value

Control state.

## Pressed (Set method)

Sets the state ("Pressed" property) of the control.

```
bool Pressed(  
    const bool pressed    // new state  
)
```

### Parameters

*pressed*

[in] New control state.

### Returned value

true if successful, otherwise false.

## Locking (Get method)

Gets the "Locking" property of the control.

```
bool Locking() const
```

### Returned value

The value of "Locking" property.

## Locking (Set method)

Sets new value of the "Locking" property of the control.

```
void Locking(  
    const bool locking    // new value  
)
```

### Parameters

*locking*

[in] New value of "Locking" property.

### Returned value

None.



## OnSetZOrder

The control "SetZOrder" (change of the [OBJPROP\\_ZORDER](#) property) event handler.

```
virtual bool OnSetZOrder()
```

### Returned value

true if the event has been processed, otherwise false.

## OnCreate

The control "Create" event handler.

```
virtual bool OnCreate()
```

### Returned value

true if the event has been processed, otherwise false.

## OnShow

The control "Show" event handler.

```
virtual bool OnShow()
```

### Returned value

true if the event has been processed, otherwise false.

## OnHide

The control "Hide" event handler.

```
virtual bool OnHide()
```

### Returned value

true if the event has been processed, otherwise false.

## OnMove

The control "Move" event handler.

```
virtual bool OnMove()
```

### Returned value

true if the event has been processed, otherwise false.

## OnChange

The control "Change" event handler.

```
virtual bool OnChange()
```

### Returned value

true if the event has been processed, otherwise false.

## OnActivate

The control "Activate" event handler.

```
virtual bool OnActivate()
```

### Returned value

true if the event has been processed, otherwise false.

## OnDeactivate

The control "Deactivate" event handler.

```
virtual bool OnDeactivate()
```

### Returned value

true if the event has been processed, otherwise false.



## OnMouseDown

The control "MouseDown" event handler.

```
virtual bool OnMouseDown()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The "MouseDown" event occurs when left mouse button is pressed on the control.

## OnMouseUp

The control "MouseUp" (left mouse button release) event handler.

```
virtual bool OnMouseUp()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The "MouseUp" event occurs when left mouse button is released on the control.

## CButton

CButton is class of the simple control, based on "Button" chart object.

### Description

CButton class is intended for creation of simple buttons.

### Declaration

```
class CButton : public CWndObj
```

### Title

```
#include <Controls\Button.mqh>
```

### Class Methods

| Create                               |                                    |
|--------------------------------------|------------------------------------|
| <a href="#">Create</a>               | Creates control                    |
| State                                |                                    |
| <a href="#">Pressed</a>              | Gets/Sets the "Pressed" property   |
| <a href="#">Locking</a>              | Gets/Sets the "Locking" property   |
| Properties change event handlers     |                                    |
| <a href="#">OnSetText</a>            | "SetText" event handler            |
| <a href="#">OnSetColor</a>           | "SetColor" event handler           |
| <a href="#">OnSetColorBackground</a> | "SetColorBackground" event handler |
| <a href="#">OnSetColorBorder</a>     | "SetColorBorder" event handler     |
| <a href="#">OnSetFont</a>            | "SetFont" event handler            |
| <a href="#">OnSetFontSize</a>        | "SetFontSize" event handler        |
| Internal event handlers              |                                    |
| <a href="#">OnCreate</a>             | "Create" event handler             |
| <a href="#">OnShow</a>               | "Show" event handler               |
| <a href="#">OnHide</a>               | "Hide" event handler               |
| <a href="#">OnMove</a>               | "Move" event handler               |
| <a href="#">OnResize</a>             | "Resize" event handler             |
| <a href="#">OnMouseDown</a>          | "MouseDown" event handler          |
| <a href="#">OnOnMouseUp</a>          | "MouseUp" event handler            |



## Create

Creates new CButton control.

```
virtual bool Create(  
    const long    chart,      // chart ID  
    const string  name,      // name  
    const int     subwin,     // chart subwindow  
    const int     x1,        // x1 coordinate  
    const int     y1,        // y1 coordinate  
    const int     x2,        // x2 coordinate  
    const int     y2         // y2 coordinate  
)
```

### Parameters

*chart*

[in] chart ID.

*name*

[in] Unique name of the control.

*subwin*

[in] Chart subwindow.

*x1*

[in] X coordinate of the upper left corner.

*y1*

[in] Y coordinate of the upper left corner.

*x2*

[in] X coordinate of the lower right corner.

*y2*

[in] Y coordinate of the lower right corner.

### Returned value

true if successful, otherwise false.

## Pressed (Get method)

Gets the state ("Pressed" property) of the control.

```
bool Pressed() const
```

### Returned value

Control state.

## Pressed (Set method)

Sets the state ("Pressed" property) of the control.

```
bool Pressed(  
    const bool pressed // new state  
)
```

### Parameters

*pressed*

[in] New control state.

### Returned value

true if successful, otherwise false.

## Locking (Get method)

Gets the "Locking" property of the control.

```
bool Locking() const
```

### Returned value

The value of "Locking" property.

## Locking (Set method)

Sets new value of "Locking" property of the control.

```
void Locking(  
    const bool locking    // new value  
)
```

### Parameters

*locking*

[in] New value of "Locking" property.

### Returned value

None.

## OnSetText

The control "SetText" (change of the [OBJPROP\\_TEXT](#) property) event handler.

```
virtual bool OnSetText()
```

### Returned value

true if the event has been processed, otherwise false.



## OnSetColor

The control "SetColor" (change of the [OBJPROP\\_COLOR](#) property) event handler.

```
virtual bool OnSetColor()
```

### Returned value

true if the event has been processed, otherwise false.

## OnSetColorBackground

The control "SetColorBackground" (change of the [OBJPROP\\_BGCOLOR](#) property) event handler.

```
virtual bool OnSetColorBackground()
```

### Returned value

true if the event has been processed, otherwise false.

## OnSetColorBorder

The control "SetColorBorder" (change of the [OBJPROP\\_BORDER\\_COLOR](#) property) event handler.

```
virtual bool OnSetColorBackground()
```

### Returned value

true if the event has been processed, otherwise false.

## OnSetFont

The control "SetFont" (change of the [OBJPROP\\_FONT](#) property) event handler.

```
virtual bool OnSetFont()
```

### Returned value

true if the event has been processed, otherwise false.

## OnSetFontSize

The control "SetFontSize" (change of the [OBJPROP\\_FONTSIZE](#) property) event handler.

```
virtual bool OnSetFontSize()
```

### Returned value

true if the event has been processed, otherwise false.

## OnCreate

The control "Create" event handler.

```
virtual bool OnCreate()
```

### Returned value

true if the event has been processed, otherwise false.

## OnShow

The control "Show" event handler.

```
virtual bool OnShow()
```

### Returned value

true if the event has been processed, otherwise false.

## OnHide

The control "Hide" event handler.

```
virtual bool OnHide()
```

### Returned value

true if the event has been processed, otherwise false.



## OnMove

The control "Move" event handler.

```
virtual bool OnMove()
```

### Returned value

true if the event has been processed, otherwise false.

## OnResize

The control "Resize" event handler.

```
virtual bool OnResize()
```

### Returned value

true if the event has been processed, otherwise false.

## OnMouseDown

The control "MouseDown" event handler.

```
virtual bool OnMouseDown()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The "MouseDown" event occurs when left mouse button is pressed on the control.

## OnMouseUp

The control "MouseUp" (left mouse button release) event handler.

```
virtual bool OnMouseUp()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The "MouseUp" event occurs when left mouse button is released on the control.

## CEdit

CEdit is class of the simple control, based on "Edit" chart object.

### Description

CEdit class is intended for creation of controls, in which the user can enter text.

### Declaration

```
class CEdit : public CWndObj
```

### Title

```
#include <Controls\Edit.mqh>
```

### Class Methods

| Create                               |                                                                       |
|--------------------------------------|-----------------------------------------------------------------------|
| <a href="#">Create</a>               | Creates control                                                       |
| Properties                           |                                                                       |
| <a href="#">ReadOnly</a>             | Gets/Sets the "ReadOnly" property                                     |
| <a href="#">TextAlign</a>            | Gets/Sets the "TextAlign" property                                    |
| Chart object event handlers          |                                                                       |
| <a href="#">OnObjectEndEdit</a>      | The <a href="#">CHARTEVENT_OBJECT_ENDEDIT</a> event handler (virtual) |
| Properties change event handlers     |                                                                       |
| <a href="#">OnSetText</a>            | "SetText" event handler                                               |
| <a href="#">OnSetColor</a>           | "SetColor" event handler                                              |
| <a href="#">OnSetColorBackground</a> | "SetColorBackground" event handler                                    |
| <a href="#">OnSetColorBorder</a>     | "SetColorBorder" event handler                                        |
| <a href="#">OnSetFont</a>            | "SetFont" event handler                                               |
| <a href="#">OnSetFontSize</a>        | "SetFontSize" event handler                                           |
| <a href="#">OnSetZOrder</a>          | "SetZOrder" event handler                                             |
| Internal event handlers              |                                                                       |
| <a href="#">OnCreate</a>             | "Create" event handler                                                |
| <a href="#">OnShow</a>               | "Show" event handler                                                  |
| <a href="#">OnHide</a>               | "Hide" event handler                                                  |
| <a href="#">OnMove</a>               | "Move" event handler                                                  |

|                                 |                        |
|---------------------------------|------------------------|
| <a href="#"><u>OnResize</u></a> | "Resize" event handler |
| <a href="#"><u>OnChange</u></a> | "Change" event handler |
| <a href="#"><u>OnClick</u></a>  | "Click" event handler  |

## Create

Creates new CEdit control.

```
virtual bool Create(  
    const long    chart,      // chart ID  
    const string  name,      // name  
    const int     subwin,     // chart subwindow  
    const int     x1,        // x1 coordinate  
    const int     y1,        // y1 coordinate  
    const int     x2,        // x2 coordinate  
    const int     y2         // y2 coordinate  
)
```

### Parameters

*chart*

[in] chart ID.

*name*

[in] Unique name of the control.

*subwin*

[in] Chart subwindow.

*x1*

[in] X coordinate of the upper left corner.

*y1*

[in] Y coordinate of the upper left corner.

*x2*

[in] X coordinate of the lower right corner.

*y2*

[in] Y coordinate of the lower right corner.

### Returned value

true if successful, otherwise false.

## ReadOnly (Get Method)

Gets the "ReadOnly" property of the control.

```
bool ReadOnly()
```

### Returned value

The value of "ReadOnly" property.

## ReadOnly (Set Method)

Sets the value of "ReadOnly" property of the control.

```
bool ReadOnly(  
    const bool flag    // new values  
)
```

### Parameters

*flag*

[in] New value of "ReadOnly" property.

### Returned value

true if successful, otherwise false.



## TextAlign (Get method)

Gets the value of "TextAlign" property ([text alignment mode](#)) of the control.

```
ENUM_ALIGN_MODE TextAlign() const
```

### Returned value

Value of "TextAlign" property of the control.

## TextAlign (Set method)

Sets new value of "TextAlign" property ([text alignment mode](#)) of the control.

```
bool TextAlign(  
    ENUM_ALIGN_MODE align    // new value  
)
```

### Parameters

*align*

[in] New value of "TextAlign" property.

### Returned value

true if successful, false if property hasn't changed.

## OnObjectEndEdit

The [CHARTEVENT\\_OBJECT\\_ENDEDIT](#) event handler.

```
virtual bool OnObjectEndEdit ()
```

### Returned value

true if the event has been processed, otherwise false.

## OnSetText

The control "SetText" (change of the [OBJPROP\\_TEXT](#) property) event handler.

```
virtual bool OnSetText()
```

### Returned value

true if the event has been processed, otherwise false.

## OnSetColor

The control "SetColor" (change of the [OBJPROP\\_COLOR](#) property) event handler.

```
virtual bool OnSetColor()
```

### Returned value

true if the event has been processed, otherwise false.

## OnSetColorBackground

The control "SetColorBackground" (change of the [OBJPROP\\_BGCOLOR](#) property) event handler.

```
virtual bool OnSetColorBackground()
```

### Returned value

true if the event has been processed, otherwise false.

## OnSetColorBorder

The control "SetColorBorder" (change of the [OBJPROP\\_BORDER\\_COLOR](#) property) event handler.

```
virtual bool OnSetColorBackground()
```

### Returned value

true if the event has been processed, otherwise false.

## OnSetFont

The control "SetFont" (change of the [OBJPROP\\_FONT](#) property) event handler.

```
virtual bool OnSetFont()
```

### Returned value

true if the event has been processed, otherwise false.

## OnSetFontSize

The control "SetFontSize" (change of the [OBJPROP\\_FONTSIZE](#) property) event handler.

```
virtual bool OnSetFontSize()
```

### Returned value

true if the event has been processed, otherwise false.



## OnSetZOrder

The control "SetZOrder" (change of the [OBJPROP\\_ZORDER](#) property) event handler.

```
virtual bool OnSetZOrder()
```

### Returned value

true if the event has been processed, otherwise false.

## OnCreate

The control "Create" event handler.

```
virtual bool OnCreate()
```

### Returned value

true if the event has been processed, otherwise false.

## OnShow

The control "Show" event handler.

```
virtual bool OnShow()
```

### Returned value

true if the event has been processed, otherwise false.

## OnHide

The control "Hide" event handler.

```
virtual bool OnHide()
```

### Returned value

true if the event has been processed, otherwise false.

## OnMove

The control "Move" event handler.

```
virtual bool OnMove()
```

### Returned value

true if the event has been processed, otherwise false.

## OnResize

The control "Resize" event handler.

```
virtual bool OnResize()
```

### Returned value

true if the event has been processed, otherwise false.

## OnChange

The control "Change" event handler.

```
virtual bool OnChange()
```

### Returned value

true if the event has been processed, otherwise false.

## OnClick

The control "Click" (left mouse button click) event handler.

```
virtual bool OnClick()
```

### Returned value

true if the event has been processed, otherwise false.



## CPanel

CPanel is class of the simple control, based on "Rectangle label" chart object.

### Description

CPanel class is intended to combine the controls with similar functions in the group.

### Declaration

```
class CPanel : public CWndObj
```

### Title

```
#include <Controls\Panel.mqh>
```

### Class Methods

| Create                               |                                                    |
|--------------------------------------|----------------------------------------------------|
| <a href="#">Create</a>               | Creates control                                    |
| Chart object properties              |                                                    |
| <a href="#">BorderType</a>           | Gets the "BorderType" property of the chart object |
| Chart object event handlers          |                                                    |
| <a href="#">OnSetText</a>            | "SetText" event handler                            |
| <a href="#">OnSetColorBackground</a> | "SetColorBackground" event handler                 |
| <a href="#">OnSetColorBorder</a>     | "SetColorBorder" event handler                     |
| Internal event handlers              |                                                    |
| <a href="#">OnCreate</a>             | "Create" event handler                             |
| <a href="#">OnShow</a>               | "Show" event handler                               |
| <a href="#">OnHide</a>               | "Hide" event handler                               |
| <a href="#">OnMove</a>               | "Move" event handler                               |
| <a href="#">OnResize</a>             | "Resize" event handler                             |
| <a href="#">OnChange</a>             | "Change" event handler                             |

## Create

Creates new CPanel control.

```
virtual bool Create(  
    const long    chart,      // chart ID  
    const string  name,      // name  
    const int     subwin,     // chart subwindow  
    const int     x1,        // x1 coordinate  
    const int     y1,        // y1 coordinate  
    const int     x2,        // x2 coordinate  
    const int     y2         // y2 coordinate  
)
```

### Parameters

*chart*

[in] chart ID.

*name*

[in] Unique name of the control.

*subwin*

[in] Chart subwindow.

*x1*

[in] X coordinate of the upper left corner.

*y1*

[in] Y coordinate of the upper left corner.

*x2*

[in] X coordinate of the lower right corner.

*y2*

[in] Y coordinate of the lower right corner.

### Returned value

true if successful, otherwise false.

## BorderType (Get method)

Gets the "BorderType" property of the chart object.

```
ENUM_BORDER_TYPE BorderType()
```

### Returned value

The value of "BorderType" property.

## BorderType (Set method)

Sets new value of "BorderType" property of the chart object.

```
bool BorderType(  
    const ENUM_BORDER_TYPE type // value  
)
```

### Parameters

*type*

[in] New value of "BorderType" property.

### Returned value

true if successful, otherwise false.

## OnSetText

The control "SetText" (change of the [OBJPROP\\_TEXT](#) property) event handler.

```
virtual bool OnSetText()
```

### Returned value

true if the event has been processed, otherwise false.

## OnSetColorBackground

The control "SetColorBackground" (change of the [OBJPROP\\_BGCOLOR](#) property) event handler.

```
virtual bool OnSetColorBackground()
```

### Returned value

true if the event has been processed, otherwise false.

## OnSetColorBorder

The control "SetColorBorder" (change of the [OBJPROP\\_BORDER\\_COLOR](#) property) event handler.

```
virtual bool OnSetColorBackground()
```

### Returned value

true if the event has been processed, otherwise false.

## OnCreate

The control "Create" event handler.

```
virtual bool OnCreate()
```

### Returned value

true if the event has been processed, otherwise false.

## OnShow

The control "Show" event handler.

```
virtual bool OnShow()
```

### Returned value

true if the event has been processed, otherwise false.



## OnHide

The control "Hide" event handler.

```
virtual bool OnHide()
```

### Returned value

true if the event has been processed, otherwise false.

## OnMove

The control "Move" event handler.

```
virtual bool OnMove()
```

### Returned value

true if the event has been processed, otherwise false.

## OnResize

The control "Resize" event handler.

```
virtual bool OnResize()
```

### Returned value

true if the event has been processed, otherwise false.

## OnChange

The control "Change" event handler.

```
virtual bool OnChange()
```

### Returned value

true if the event has been processed, otherwise false.

## CPicture

CPicture is class of the simple control, based on "Bitmap Label" chart object.

### Description

CPicture class is intended for creation of simple graphic images.

### Declaration

```
class CPicture : public CWndObj
```

### Title

```
#include <Controls\Picture.mqh>
```

### Class Methods

| Create                   |                                                |
|--------------------------|------------------------------------------------|
| <a href="#">Create</a>   | Creates control                                |
| Chart object properties  |                                                |
| <a href="#">Border</a>   | Gets/Sets the border width of the chart object |
| <a href="#">BmpName</a>  | Gets/Sets the name of bmp-file of the control  |
| Internal events          |                                                |
| <a href="#">OnCreate</a> | "Create" event handler                         |
| <a href="#">OnShow</a>   | "Show" event handler                           |
| <a href="#">OnHide</a>   | "Hide" event handler                           |
| <a href="#">OnMove</a>   | "Move" event handler                           |
| <a href="#">OnChange</a> | "Change" event handler                         |

## Create

Creates new CPicture control.

```
virtual bool Create(  
    const long    chart,      // chart ID  
    const string  name,      // name  
    const int     subwin,     // chart subwindow  
    const int     x1,        // x1 coordinate  
    const int     y1,        // y1 coordinate  
    const int     x2,        // x2 coordinate  
    const int     y2         // y2 coordinate  
)
```

### Parameters

*chart*

[in] chart ID.

*name*

[in] Unique name of the control.

*subwin*

[in] Chart subwindow.

*x1*

[in] X coordinate of the upper left corner.

*y1*

[in] Y coordinate of the upper left corner.

*x2*

[in] X coordinate of the lower right corner.

*y2*

[in] Y coordinate of the lower right corner.

### Returned value

true if successful, otherwise false.

## Border (Get method)

Gets the "Border" (border width) property of the control.

```
int Border() const
```

### Returned value

The "Border" property.

## Border (Set method)

Sets the "Border" (border width) property of the control.

```
bool Border(  
    const int value    // new value  
)
```

### Parameters

*value*

[in] New value of the "Border" property.

### Returned value

true if successful, otherwise false.

## BmpName (Get method)

Gets the name of bmp file of the control.

```
string BmpName() const
```

### Returned value

Name of bmp file of the control.

## BmpName (Set method)

Sets the name of bmp file of the control.

```
bool BmpName (
    const string name    // file name
)
```

### Parameters

*name*

[in] Name of bmp file of the control.

### Returned value

true if successful, otherwise false.



## OnCreate

The control "Create" event handler.

```
virtual bool OnCreate()
```

### Returned value

true if the event has been processed, otherwise false.

## OnShow

The control "Show" event handler.

```
virtual bool OnShow()
```

### Returned value

true if the event has been processed, otherwise false.

## OnHide

The control "Hide" event handler.

```
virtual bool OnHide()
```

### Returned value

true if the event has been processed, otherwise false.

## OnMove

The control "Move" event handler.

```
virtual bool OnMove()
```

### Returned value

true if the event has been processed, otherwise false.

## OnChange

The control "Change" event handler.

```
virtual bool OnChange()
```

### Returned value

true if the event has been processed, otherwise false.

## CScroll

CScroll is a base class for creation of scroll bars.

### Description

CScroll is a complex control (with dependent controls), it contains the base functionality for creation of scroll bars. The base class itself isn't used as a separate control, two of its heirs ([CScrollV](#) and [CScrollH](#) classes) are used as controls.

### Declaration

```
class CScroll : public CWndContainer
```

### Title

```
#include <Controls\Scrolls.mqh>
```

### Class Methods

| Create                            |                                                             |
|-----------------------------------|-------------------------------------------------------------|
| <a href="#">Create</a>            | Creates control                                             |
| Chart object event handlers       |                                                             |
| <a href="#">OnEvent</a>           | Event handler of all chart events                           |
| Properties                        |                                                             |
| <a href="#">MinPos</a>            | Gets/Sets the minimal position                              |
| <a href="#">MaxPos</a>            | Gets/Sets the maximal position                              |
| <a href="#">CurrPos</a>           | Gets/Sets the current position                              |
| Dependent controls creation       |                                                             |
| <a href="#">CreateBack</a>        | Creates background button                                   |
| <a href="#">CreateInc</a>         | Creates increment button of the scroll bar                  |
| <a href="#">CreateDec</a>         | Creates decrement button of the scroll bar                  |
| <a href="#">CreateThumb</a>       | Creates thumb button (can be dragged) of the scroll bar     |
| Dependent controls event handlers |                                                             |
| <a href="#">OnClickInc</a>        | Event handler, used for handling of increment button events |
| <a href="#">OnClickDec</a>        | Event handler, used for handling of decrement button events |
| Internal event handlers           |                                                             |
| <a href="#">OnShow</a>            | "Create" event handler                                      |

|                                    |                                        |
|------------------------------------|----------------------------------------|
| <a href="#">OnHide</a>             | "Hide" event handler                   |
| <a href="#">OnChangePos</a>        | "ChangePosition" event handler         |
| <b>Object drag handlers</b>        |                                        |
| <a href="#">OnThumbDragStart</a>   | "ThumbDragStart" event handler         |
| <a href="#">OnThumbDragProcess</a> | "ThumbDragProcess" event handler       |
| <a href="#">OnThumbDragEnd</a>     | "ThumbDragEnd" event handler           |
| <b>Position</b>                    |                                        |
| <a href="#">CalcPos</a>            | Gets scroll bar position by coordinate |

## Create

Creates new CScroll control.

```
virtual bool Create(  
    const long    chart,      // chart ID  
    const string  name,      // name  
    const int     subwin,     // chart subwindow  
    const int     x1,        // x1 coordinate  
    const int     y1,        // y1 coordinate  
    const int     x2,        // x2 coordinate  
    const int     y2         // y2 coordinate  
)
```

### Parameters

*chart*

[in] chart ID.

*name*

[in] Unique name of the control.

*subwin*

[in] Chart subwindow.

*x1*

[in] X coordinate of the upper left corner.

*y1*

[in] Y coordinate of the upper left corner.

*x2*

[in] X coordinate of the lower right corner.

*y2*

[in] Y coordinate of the lower right corner.

### Returned value

true if successful, otherwise false.



## OnEvent

Chart event handler.

```
virtual bool OnEvent(  
    const int      id,           // ID  
    const long&    lparam,       // event parameter of long type  
    const double&  dparam,       // event parameter of double type  
    const string&  sparam        // event parameter of string type  
)
```

### Parameters

*id*

[in] Event ID.

*lparam*

[in] Event parameter of [long](#) type, passed by reference.

*dparam*

[in] Event parameter of [double](#) type, passed by reference.

*sparam*

[in] Event parameter of [string](#) type, passed by reference.

### Returned value

true - if the event has been processed, otherwise false.

## MinPos (Get method)

Gets the value of "MinPos" (minimal position) of the CScroll control.

```
int MinPos() const
```

### Returned value

New value of "MinPos" property.

## MinPos (Set method)

Sets the value of "MinPos" (minimal position) of the CScroll control.

```
void MinPos(  
    const int value    // new value  
)
```

### Parameters

*value*

[in] New value of "MinPos" property.

### Returned value

None.

## MaxPos (Get method)

Gets the value of "MinPos" (maximal position) of the CScroll control.

```
int MaxPos() const
```

### Returned value

New value of "MaxPos" property.

## MaxPos (Set method)

Sets the value of "MaxPos" (maximal position) of the CScroll control.

```
void MaxPos(  
    const int value    // new value  
)
```

### Parameters

*value*

[in] New value of "MaxPos" property.

### Returned value

None.

## CurrPos (Get method)

Gets the value of "CurrPos" (current position) of the CScroll control.

```
int CurrPos() const
```

### Returned value

New value of "MaxPos" property.

## CurrPos (Set method)

Sets the value of "CurrPos" (current position) of the CScroll control.

```
void CurrPos(  
    const int value    // new value  
)
```

### Parameters

*value*

[in] New value of "CurrPos" property.

### Returned value

None.

## CreateBack

Creates background button of the CScroll control.

```
virtual bool CreateBack()
```

### Returned value

true if successful, otherwise false.

## CreateInc

Creates increment button of the CScroll control.

```
virtual bool CreateInc()
```

### Returned value

true if successful, otherwise false.

## CreateDec

Creates decrement button of CScroll control.

```
virtual bool CreateDec()
```

### Returned value

true if successful, otherwise false.

## CreateThumb

Creates thumb button (can be dragged) of the CScroll control.

```
virtual bool CreateThumb()
```

### Returned value

true if successful, otherwise false.



## OnClickInc

The control "ClickInc" (left mouse button click on the increment button) event handler.

```
virtual bool OnClickInc()
```

### Returned value

true if the event has been processed, otherwise false.

## OnClickDec

The control "ClickDec" (left mouse button click on the decrement button) event handler.

```
virtual bool OnClickDec()
```

### Returned value

true if the event has been processed, otherwise false.

## OnShow

The control "Show" event handler.

```
virtual bool OnShow()
```

### Returned value

true if the event has been processed, otherwise false.

## OnHide

The control "Hide" event handler.

```
virtual bool OnHide()
```

### Returned value

true if the event has been processed, otherwise false.

## OnChangePos

The control "ChangePos" (position change) event handler.

```
virtual bool OnChangePos()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The base class method does nothing and always returns true.

## OnThumbDragStart

The control "ThumbDragStart" (drag start) event handler.

```
virtual bool OnThumbDragStart ()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The "ThumbDragStart" event occurs at start of the drag operation.

## OnThumbDragProcess

The control "ThumbDragProcess" event handler.

```
virtual bool OnThumbDragProcess(  
    const int x,      // x coordinate  
    const int y      // y coordinate  
)
```

### Parameters

*x*

[in] Current X coordinate of mouse cursor.

*y*

[in] Current Y coordinate of mouse cursor.

### Returned value

true if the event has been processed, otherwise false.

### Note

The "ThumbDragProcess" occurs when the scroll bar control (thumb button) is moved.

## OnThumbDragEnd

The control "ThumbDragEnd" (drag process finished) event handler.

```
virtual bool OnThumbDragEnd()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The "ThumbDragEnd" occurs when the drag operation of the scroll bar control (thumb button) is finished.



## CalcPos

Gets scroll bar position by coordinate.

```
virtual int CalcPos(  
    const int coord // coordinate  
)
```

### Parameters

*coord*

[in] Scroll bar coordinate.

### Returned value

Scroll bar position.

## CScrollV

CScrollV is a class of the "Vertical scroll bar" complex control.

### Description

CScrollV class is intended for creation of vertical scroll bars.

### Declaration

```
class CScrollV : public CScroll
```

### Title

```
#include <Controls\Scrolls.mqh>
```

### Class Methods

| Dependent controls                 |                                                  |
|------------------------------------|--------------------------------------------------|
| <a href="#">CreateInc</a>          | Creates scroll bar increment button              |
| <a href="#">CreateDec</a>          | Creates scroll bar decrement button              |
| <a href="#">CreateThumb</a>        | Creates scroll bar thumb button (can be dragged) |
| Internal event handlers            |                                                  |
| <a href="#">OnResize</a>           | "Resize" event handler                           |
| <a href="#">OnChangePos</a>        | "ChangePosition" event handler                   |
| Drag event handlers                |                                                  |
| <a href="#">OnThumbDragStart</a>   | "ThumbDragStart" event handler                   |
| <a href="#">OnThumbDragProcess</a> | "ThumbDragProcess" event handler                 |
| <a href="#">OnThumbDragEnd</a>     | "ThumbDragEnd" event handler                     |
| Position                           |                                                  |
| <a href="#">CalcPos</a>            | Gets scroll bar position by coordinate           |

## CreateInc

Creates increment button of the control.

```
virtual bool CreateInc()
```

### Returned value

true if successful, otherwise false.

## CreateDec

Creates decrement button of the control.

```
virtual bool CreateDec()
```

### Returned value

true if successful, otherwise false.

## CreateThumb

Creates thumb button (can be dragged) of the control.

```
virtual bool CreateThumb()
```

### Returned value

true if successful, otherwise false.

## OnResize

The control "Resize" event handler.

```
virtual bool OnResize()
```

### Returned value

true if the event has been processed, otherwise false.

## OnChangePos

The control "ChangePos" (position change) event handler.

```
virtual bool OnChangePos()
```

### Returned value

true if the event has been processed, otherwise false.

## OnThumbDragStart

The control "ThumbDragStart" (drag start) event handler.

```
virtual bool OnThumbDragStart ()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The "ThumbDragStart" event occurs at start of the drag operation.



## OnThumbDragProcess

The control "ThumbDragProcess" event handler.

```
virtual bool OnThumbDragProcess(  
    const int x,      // x coordinate  
    const int y      // y coordinate  
)
```

### Parameters

*x*

[in] Current X coordinate of mouse cursor.

*y*

[in] Current Y coordinate of mouse cursor.

### Returned value

true if the event has been processed, otherwise false.

### Note

The "ThumbDragProcess" occurs when the scroll bar control (thumb button) is moved.

## OnThumbDragEnd

The control "ThumbDragEnd" (drag process finished) event handler.

```
virtual bool OnThumbDragEnd()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The "ThumbDragEnd" occurs when the drag operation of the scroll bar control (thumb button) is finished.

## CalcPos

Gets scroll bar position by coordinate.

```
virtual int CalcPos(  
    const int coord // coordinate  
)
```

### Parameters

*coord*

[in] Scroll bar coordinate.

### Returned value

Scroll bar position.

## CScrollH

CScrollH is a class of the "Horizontal scroll bar" complex control.

### Description

CScrollH is intended for creation of horizontal scroll bars.

### Declaration

```
class CScrollH : public CScroll
```

### Title

```
#include <Controls\Scrolls.mqh>
```

### Class Methods

| Dependent controls                 |                                                  |
|------------------------------------|--------------------------------------------------|
| <a href="#">CreateInc</a>          | Creates scroll bar increment button              |
| <a href="#">CreateDec</a>          | Creates scroll bar decrement button              |
| <a href="#">CreateThumb</a>        | Creates scroll bar thumb button (can be dragged) |
| Internal event handlers            |                                                  |
| <a href="#">OnResize</a>           | "Resize" event handler                           |
| <a href="#">OnChangePos</a>        | "ChangePosition" event handler                   |
| Drag event handlers                |                                                  |
| <a href="#">OnThumbDragStart</a>   | "ThumbDragStart" event handler                   |
| <a href="#">OnThumbDragProcess</a> | "ThumbDragProcess" event handler                 |
| <a href="#">OnThumbDragEnd</a>     | "ThumbDragEnd" event handler                     |
| Position                           |                                                  |
| <a href="#">CalcPos</a>            | Gets scroll bar position by coordinate           |

## CreateInc

Creates increment button of the control.

```
virtual bool CreateInc()
```

### Returned value

true if successful, otherwise false.

## CreateDec

Creates decrement button of the control.

```
virtual bool CreateDec()
```

### Returned value

true if successful, otherwise false.

## CreateThumb

Creates thumb button (can be dragged) of the control.

```
virtual bool CreateThumb()
```

### Returned value

true if successful, otherwise false.

## OnResize

The control "Resize" event handler.

```
virtual bool OnResize()
```

### Returned value

true if the event has been processed, otherwise false.



## OnChangePos

The control "ChangePos" (position change) event handler.

```
virtual bool OnChangePos()
```

### Returned value

true if the event has been processed, otherwise false.

## OnThumbDragStart

The control "ThumbDragStart" (drag start) event handler.

```
virtual bool OnThumbDragStart ()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The "ThumbDragStart" event occurs at start of the drag operation.

## OnThumbDragProcess

The control "ThumbDragProcess" event handler.

```
virtual bool OnThumbDragProcess(  
    const int x,      // x coordinate  
    const int y      // y coordinate  
)
```

### Parameters

*x*

[in] Current X coordinate of mouse cursor.

*y*

[in] Current Y coordinate of mouse cursor.

### Returned value

true if the event has been processed, otherwise false.

### Note

The "ThumbDragProcess" occurs when the scroll bar control (thumb button) is moved.

## OnThumbDragEnd

The control "ThumbDragEnd" (drag process finished) event handler.

```
virtual bool OnThumbDragEnd()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The "ThumbDragEnd" occurs when the drag operation of the scroll bar control (thumb button) is finished.

## CalcPos

Gets scroll bar position by coordinate.

```
virtual int CalcPos(  
    const int coord // coordinate  
)
```

### Parameters

*coord*

[in] Scroll bar coordinate.

### Returned value

Scroll bar position.

## CWndClient

CWndClient is a class of the "Client area" complex control (with dependent controls). It's a base class for creation of scroll bars area.

### Description

CWndClient implements the functionality for creation of client area with scroll bars.

### Declaration

```
class CWndClient : public CWndContainer
```

### Title

```
#include <Controls\WndClient.mqh>
```

### Class Methods

| Create                            |                                                                   |
|-----------------------------------|-------------------------------------------------------------------|
| <a href="#">Create</a>            | Creates control                                                   |
| Chart event handler               |                                                                   |
| <a href="#">OnEvent</a>           | Event handler of all chart events                                 |
| Properties                        |                                                                   |
| <a href="#">ColorBackground</a>   | Sets background color                                             |
| <a href="#">ColorBorder</a>       | Sets border color                                                 |
| <a href="#">BorderType</a>        | Sets border type                                                  |
| Settings                          |                                                                   |
| <a href="#">VScrolled</a>         | Gets/Sets the flag, indicating that vertical scroll bar is used   |
| <a href="#">HScrolled</a>         | Gets/Sets the flag, indicating that horizontal scroll bar is used |
| Dependent controls                |                                                                   |
| <a href="#">CreateBack</a>        | Creates background for scroll bar                                 |
| <a href="#">CreateScrollV</a>     | Creates vertical scroll bar                                       |
| <a href="#">CreateScrollH</a>     | Creates horizontal scroll bar                                     |
| Internal event handlers           |                                                                   |
| <a href="#">OnResize</a>          | "Resize" event handler                                            |
| Dependent controls event handlers |                                                                   |
| <a href="#">OnVScrollShow</a>     | "Show" event handler (virtual) of VScroll dependent control       |

|                                          |                                                                        |
|------------------------------------------|------------------------------------------------------------------------|
| <a href="#"><u>OnVScrollHide</u></a>     | "Hide" event handler (virtual) of VScroll dependent control            |
| <a href="#"><u>OnHScrollShow</u></a>     | "Show" event handler (virtual) of HScroll dependent control            |
| <a href="#"><u>OnHScrollHide</u></a>     | "Hide" event handler (virtual) of HScroll dependent control            |
| <a href="#"><u>OnScrollLineDown</u></a>  | "ScrollLineDown" event handler (virtual) of VScroll dependent control  |
| <a href="#"><u>OnScrollLineUp</u></a>    | "ScrollLineUp" event handler (virtual) of VScroll dependent control    |
| <a href="#"><u>OnScrollLineLeft</u></a>  | "ScrollLineLeft" event handler (virtual) of HScroll dependent control  |
| <a href="#"><u>OnScrollLineRight</u></a> | "ScrollLineRight" event handler (virtual) of HScroll dependent control |
| <b>Resize</b>                            |                                                                        |
| <a href="#"><u>Rebound</u></a>           | Sets new coordinates of the control using CRect class coordinates      |

## Create

Creates new CWndClient control.

```
virtual bool Create(  
    const long    chart,      // chart ID  
    const string  name,      // name  
    const int     subwin,     // chart subwindow  
    const int     x1,        // x1 coordinate  
    const int     y1,        // y1 coordinate  
    const int     x2,        // x2 coordinate  
    const int     y2         // y2 coordinate  
)
```

### Parameters

*chart*

[in] chart ID.

*name*

[in] Unique name of the control.

*subwin*

[in] Chart subwindow.

*x1*

[in] X coordinate of the upper left corner.

*y1*

[in] Y coordinate of the upper left corner.

*x2*

[in] X coordinate of the lower right corner.

*y2*

[in] Y coordinate of the lower right corner.

### Returned value

true if successful, otherwise false.



## OnEvent

Chart event handler.

```
virtual bool OnEvent(  
    const int      id,           // ID  
    const long&    lparam,       // event parameter of long type  
    const double&  dparam,       // event parameter of double type  
    const string&  sparam        // event parameter of string type  
)
```

### Parameters

*id*

[in] Event ID.

*lparam*

[in] Event parameter of [long](#) type, passed by reference.

*dparam*

[in] Event parameter of [double](#) type, passed by reference.

*sparam*

[in] Event parameter of [string](#) type, passed by reference.

### Returned value

true - if the event has been processed, otherwise false.

## ColorBackground

Sets background color of the control.

```
bool ColorBackground(  
    const color value    // new color  
)
```

### Parameters

*value*

[in] New background color of the control.

### Returned value

true if successful, otherwise false.

## ColorBorder

Set border color of the control.

```
bool ColorBorder(  
    const color value    // color  
)
```

### Parameters

*value*

[in] New border color of the control.

### Returned value

true if successful, otherwise false.

## BorderStyle

Sets border type of the control.

```
bool BorderType(  
    const ENUM_BORDER_TYPE type // border type  
)
```

### Parameters

*type*

[in] Border type of the control.

### Returned value

true if successful, otherwise false.

## VScrolled (Get method)

Gets the flag, indicating that vertical scroll bar is used.

```
bool VScrolled()
```

### Returned value

true, if vertical scroll bar is used, otherwise false.

## VScrolled (Set method)

Sets the flag, indicating that vertical scroll bar is used.

```
bool VScrolled(  
    const bool flag    // flag  
)
```

### Parameters

*flag*

[in] Flag.

### Returned value

true if successful, otherwise false.

## HScrolled (Get method)

Gets the flag, indicating that horizontal scroll bar is used.

```
bool HScrolled()
```

### Returned value

true, if horizontal scroll bar is used, otherwise false.

## HScrolled (Set method)

Sets the flag, indicating that horizontal scroll bar is used.

```
bool HScrolled(  
    const bool flag    // flag  
)
```

### Parameters

*flag*

[in] Flag.

### Returned value

true if successful, otherwise false.

## CreateBack

Creates background button of the control.

```
virtual bool CreateBack()
```

### Returned value

true if successful, otherwise false.

## CreateScrollV

Creates vertical scroll bar.

```
virtual bool CreateScrollV()
```

### Returned value

true if successful, otherwise false.



## CreateScrollH

Creates horizontal scroll bar.

```
virtual bool CreateScrollH()
```

### Returned value

true if successful, otherwise false.

## OnResize

The control "Resize" event handler.

```
virtual bool OnResize()
```

### Returned value

true if the event has been processed, otherwise false.

## OnVScrollShow

The control "VScrollShow" (vertical scroll bar show) event handler.

```
virtual bool OnVScrollShow()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The base class method does nothing and always returns true.

## OnVScrollHide

The control "VScrollHide" (vertical scroll bar hide) event handler.

```
virtual bool OnVScrollHide()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The base class method does nothing and always returns true.

## OnHScrollShow

The control "HScrollShow" (horizontal scroll bar show) event handler.

```
virtual bool OnHScrollShow()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The base class method does nothing and always returns true.

## OnHScrollHide

The control "HScrollHide" (vertical scroll bar hide) event handler.

```
virtual bool OnHScrollHide()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The base class method does nothing and always returns true.

## OnScrollLineDown

The control "ScrollLineDown" (vertical scroll line down) event handler.

```
virtual bool OnScrollLineDown ()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The base class method does nothing and always returns true.

## OnScrollLineUp

The control "ScrollLineUp" (vertical scroll line up) event handler.

```
virtual bool OnScrollLineUp()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The base class method does nothing and always returns true.



## OnScrollLineLeft

The control "ScrollLineLeft" (horizontal scroll line left) event handler.

```
virtual bool OnScrollLineLeft ()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The base class method does nothing and always returns true.

## OnScrollLineRight

The control "ScrollLineRight" (horizontal scroll line right) event handler.

```
virtual bool OnScrollLineRight()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The base class method does nothing and always returns true.

## ReBound

Sets new coordinates of the control using CRect class coordinates.

```
void ReBound(  
    const & CRect rect    // CRect class  
)
```

### Returned value

None.

## CListView

CListView is a class of the ListView complex control (with dependent controls).

### Description

CListView class encapsulates list-control functionality.

### Declaration

```
class CListView : public CWndClient
```

### Title

```
#include <Controls\ListView.mqh>
```

### Class Methods

| Create                            |                                                             |
|-----------------------------------|-------------------------------------------------------------|
| <a href="#">Create</a>            | Creates control                                             |
| Chart event handlers              |                                                             |
| <a href="#">OnEvent</a>           | Event handler of all chart events                           |
| Settings                          |                                                             |
| <a href="#">TotalView</a>         | Sets the number of items, shown on the control              |
| Add/Delete                        |                                                             |
| <a href="#">AddItem</a>           | Adds an item                                                |
| Data                              |                                                             |
| <a href="#">Select</a>            | Selects current list item by index                          |
| <a href="#">SelectByText</a>      | Selects current list item by text                           |
| <a href="#">SelectByValue</a>     | Selects current list item by value                          |
| Read-only data                    |                                                             |
| <a href="#">Value</a>             | Gets the value of current list item                         |
| Dependent controls                |                                                             |
| <a href="#">CreateRow</a>         | Creates a row of the ListView                               |
| Internal event handlers           |                                                             |
| <a href="#">OnResize</a>          | "Resize" event handler (virtual)                            |
| Dependent controls event handlers |                                                             |
| <a href="#">OnVScrollShow</a>     | "Show" event handler (virtual) of VScroll dependent control |

|                                         |                                                                       |
|-----------------------------------------|-----------------------------------------------------------------------|
| <a href="#"><u>OnVScrollHide</u></a>    | "Hide" event handler (virtual) of VScroll dependent control           |
| <a href="#"><u>OnScrollLineDown</u></a> | "ScrollLineDown" event handler (virtual) of VScroll dependent control |
| <a href="#"><u>OnScrollLineUp</u></a>   | "ScrollLineUp" event handler (virtual) of VScroll dependent control   |
| <a href="#"><u>OnItemClick</u></a>      | "ItemClick" event handler (virtual)                                   |
| <b>Redraw</b>                           |                                                                       |
| <a href="#"><u>Redraw</u></a>           | Redraws the control                                                   |
| <a href="#"><u>RowState</u></a>         | Sets the state of the specified row                                   |
| <a href="#"><u>CheckView</u></a>        | Checks the "visibility" of the specified row                          |

## Create

Creates new CListView control.

```
virtual bool Create(  
    const long    chart,      // chart ID  
    const string  name,      // name  
    const int     subwin,     // chart subwindow  
    const int     x1,        // x1 coordinate  
    const int     y1,        // y1 coordinate  
    const int     x2,        // x2 coordinate  
    const int     y2         // y2 coordinate  
)
```

### Parameters

*chart*

[in] chart ID.

*name*

[in] Unique name of the control.

*subwin*

[in] Chart subwindow.

*x1*

[in] X coordinate of the upper left corner.

*y1*

[in] Y coordinate of the upper left corner.

*x2*

[in] X coordinate of the lower right corner.

*y2*

[in] Y coordinate of the lower right corner.

### Returned value

true if successful, otherwise false.

## OnEvent

Chart event handler.

```
virtual bool OnEvent(  
    const int      id,           // ID  
    const long&    lparam,       // event parameter of long type  
    const double&  dparam,       // event parameter of double type  
    const string&  sparam        // event parameter of string type  
)
```

### Parameters

*id*

[in] Event ID.

*lparam*

[in] Event parameter of [long](#) type, passed by reference.

*dparam*

[in] Event parameter of [double](#) type, passed by reference.

*sparam*

[in] Event parameter of [string](#) type, passed by reference.

### Returned value

true - if the event has been processed, otherwise false.

## TotalView

Sets the number of items, shown on the control.

```
bool TotalView(  
    const int value    // items shown  
)
```

### Parameters

*value*

[in] The number of items, shown on the control.

### Returned value

true if successful, otherwise false.

### Note

The number of shown items can be specified only at once.



## AddItem

Adds an item to the control.

```
bool AddItem(  
    const string item,      // text  
    const long  value      // value  
)
```

### Parameters

*item*

[in] Text.

*value*

[in] Value of [long](#) type.

### Returned value

true if successful, otherwise false.

## Select

Selects current list item by index.

```
bool Select(  
    const int index    // index  
)
```

### Parameters

*index*

[in] Item index.

### Returned value

true if successful, otherwise false.

## SelectByText

Selects current list item by text.

```
bool SelectByText(  
    const string text    // text  
)
```

### Parameters

*text*

[in] Text.

### Returned value

true if successful, otherwise false.

## SelectByValue

Selects current list item by value.

```
bool SelectByValue(  
    const long value    // value  
)
```

### Parameters

*value*

[in] Value of [long](#) type.

### Returned value

true if successful, otherwise false.

## Value

Gets the value of current list item.

```
long Value()
```

### Returned value

The value of current list item.

## CreateRow

Creates a row of the CListView control.

```
bool CreateRow(  
    const int index    // index  
)
```

### Parameters

*index*

[in] Item index.

### Returned value

true if successful, otherwise false.

## OnResize

The control "Resize" event handler.

```
virtual bool OnResize()
```

### Returned value

true if the event has been processed, otherwise false.

## OnVScrollShow

The control "VScrollShow" (vertical scroll bar show) event handler.

```
virtual bool OnVScrollShow()
```

### Returned value

true if the event has been processed, otherwise false.



## OnVScrollHide

The control "VScrollHide" (vertical scroll bar hide) event handler.

```
virtual bool OnVScrollHide()
```

### Returned value

true if the event has been processed, otherwise false.

## OnScrollLineDown

The control "ScrollLineDown" (vertical scroll line down) event handler.

```
virtual bool OnScrollLineDown ()
```

### Returned value

true if the event has been processed, otherwise false.

## OnScrollLineUp

The control "ScrollLineUp" (vertical scroll line up) event handler.

```
virtual bool OnScrollLineUp()
```

### Returned value

true if the event has been processed, otherwise false.

## OnItemClick

The control "ItemClick" (mouse button click on a row) event handler.

```
virtual bool OnItemClick()  
    const int    index    // row index  
    )
```

### Returned value

true if the event has been processed, otherwise false.

## Redraw

Redraws the control.

```
bool Redraw()
```

### Returned value

true if successful, otherwise false.

## RowState

Sets the state of the specified row.

```
bool RowState(  
    const int   index    // index  
    const bool  select    // state  
)
```

### Parameters

*index*

[in] Row index.

*select*

[in] Row state.

### Returned value

true if successful, otherwise false.

## CheckView

Checks the "visibility" of the specified row.

```
bool CheckView()
```

### Returned value

true, if the selected row is visible, othewise false.

## CComboBox

CComboBox is a class of the ComboBox complex control (with dependent controls).

### Description

ComboBox consists of a list box, combined with a static control, intended for selection. The list-box portion of the control may be drop down when the user selects the drop-down arrow next to the control

### Declaration

```
class CComboBox : public CWndContainer
```

### Title

```
#include <Controls\ComboBox.mqh>
```

### Class Methods

| Create                            |                                                |
|-----------------------------------|------------------------------------------------|
| <a href="#">Create</a>            | Creates control                                |
| Chart event handlers              |                                                |
| <a href="#">OnEvent</a>           | Event handler of all chart events              |
| Add                               |                                                |
| <a href="#">AddItem</a>           | Adds an item                                   |
| Settings                          |                                                |
| <a href="#">ListViewItems</a>     | Sets the number of items, shown on the control |
| Data                              |                                                |
| <a href="#">Select</a>            | Selects current list item by index             |
| <a href="#">SelectByText</a>      | Selects current list item by text              |
| <a href="#">SelectByValue</a>     | Selects current list item by value             |
| Read-only data                    |                                                |
| <a href="#">Value</a>             | Gets the value of the current list item        |
| Dependent controls                |                                                |
| <a href="#">CreateEdit</a>        | Creates dependent control (edit)               |
| <a href="#">CreateButton</a>      | Creates dependent control (button)             |
| <a href="#">CreateList</a>        | Creates dependent control (list view)          |
| Dependent controls event handlers |                                                |
| <a href="#">OnClickEdit</a>       | "ClickEdit" event handler (virtual)            |



|                               |                                       |
|-------------------------------|---------------------------------------|
| <a href="#">OnClickButton</a> | "ClickButton" event handler (virtual) |
| <a href="#">OnChangeList</a>  | "ChangeList" event handler (virtual)  |
| <b>Show/Hide</b>              |                                       |
| <a href="#">ListShow</a>      | Shows the items list                  |
| <a href="#">ListHide</a>      | Hides the items list                  |

## Create

Creates new CComboBox control.

```
virtual bool Create(  
    const long    chart,      // chart ID  
    const string  name,      // name  
    const int     subwin,     // chart subwindow  
    const int     x1,        // x1 coordinate  
    const int     y1,        // y1 coordinate  
    const int     x2,        // x2 coordinate  
    const int     y2         // y2 coordinate  
)
```

### Parameters

*chart*

[in] chart ID.

*name*

[in] Unique name of the control.

*subwin*

[in] Chart subwindow.

*x1*

[in] X coordinate of the upper left corner.

*y1*

[in] Y coordinate of the upper left corner.

*x2*

[in] X coordinate of the lower right corner.

*y2*

[in] Y coordinate of the lower right corner.

### Returned value

true if successful, otherwise false.

## OnEvent

Chart event handler.

```
virtual bool OnEvent(  
    const int      id,           // ID  
    const long&    lparam,       // event parameter of long type  
    const double&  dparam,       // event parameter of double type  
    const string&  sparam        // event parameter of string type  
)
```

### Parameters

*id*

[in] Event ID.

*lparam*

[in] Event parameter of [long](#) type, passed by reference.

*dparam*

[in] Event parameter of [double](#) type, passed by reference.

*sparam*

[in] Event parameter of [string](#) type, passed by reference.

### Returned value

true - if the event has been processed, otherwise false.

## AddItem

Adds an item to the control.

```
bool AddItem(  
    const string item,      // text  
    const long  value      // value  
)
```

### Parameters

*item*

[in] Text.

*value=0*

[in] Value of [long](#) type.

### Returned value

true if successful, otherwise false.

## ListViewItems

Sets the number of list items of the CComboBox control.

```
void ListViewItems(  
    const int    value    // number of list items  
)
```

### Parameters

*value*

[in] Number of list items.

### Returned value

true if successful, otherwise false.

## Select

Selects current list item by index.

```
bool Select(  
    const int index    // index  
)
```

### Parameters

*index*

[in] Item index.

### Returned value

true if successful, otherwise false.

## SelectByText

Selects current list item by text.

```
bool SelectByText(  
    const string text    // text  
)
```

### Parameters

*text*

[in] Text of the item.

### Returned value

true if successful, otherwise false.

## SelectByValue

Selects current list item by value.

```
bool SelectByValue(  
    const long value    // value  
)
```

### Parameters

*value*

[in] Value of [long](#) type.

### Returned value

true if successful, otherwise false.



## Value

Gets the value of the current list item.

```
long Value()
```

### Returned value

The value of the current list item.

## CreateEdit

Creates dependent control (edit) of the control.

```
virtual bool CreateEdit()
```

### Returned value

true if successful, otherwise false.

## CreateButton

Creates dependent control (button).

```
virtual bool CreateButton()
```

### Returned value

true if successful, otherwise false.

## CreateList

Creates dependent control (list view).

```
virtual bool CreateList()
```

### Returned value

true if successful, otherwise false.

## OnClickEdit

The control "ClickEdit" (mouse click on the edit) event handler.

```
virtual bool OnClickEdit()
```

### Returned value

true if the event has been processed, otherwise false.

## OnClickButton

The control "ClickButton" (mouse click on the button) event handler.

```
virtual bool OnClickButton()
```

### Returned value

true if the event has been processed, otherwise false.

## OnChangeList

The "ChangeList" (change of the list) event handler.

```
virtual bool OnChangeList ()
```

### Returned value

true if the event has been processed, otherwise false.

## ListShow

Shows the items list.

```
virtual bool ListShow()
```

### Returned value

true if successful, otherwise false.



## ListHide

Hides the items list.

```
virtual bool ListHide()
```

### Returned value

true if successful, otherwise false.

## CCheckBox

CCheckBox is class of the CheckBox complex control.

### Description

CCheckBox control displays a check box that allows the user to select a true or false condition.

### Declaration

```
class CCheckBox : public CWndContainer
```

### Title

```
#include <Controls\CheckBox.mqh>
```

### Class Methods

| Create                            |                                                             |
|-----------------------------------|-------------------------------------------------------------|
| <a href="#">Create</a>            | Creates control                                             |
| Chart event handlers              |                                                             |
| <a href="#">OnEvent</a>           | Event handler of all chart events                           |
| Properties                        |                                                             |
| <a href="#">Text</a>              | Gets/Sets text label, associated with the control           |
| <a href="#">Color</a>             | Gets/Sets color of text label, associated with the control  |
| State                             |                                                             |
| <a href="#">Checked</a>           | Gets/Sets a value indicating whether the control is checked |
| Data                              |                                                             |
| <a href="#">Value</a>             | Gets/Sets the value associated with the control             |
| Dependent controls                |                                                             |
| <a href="#">CreateButton</a>      | Creates dependent control (button)                          |
| <a href="#">CreateLabel</a>       | Creates dependent control (label)                           |
| Dependent controls event handlers |                                                             |
| <a href="#">ClickButton</a>       | "ClickButton" event handler (virtual)                       |
| <a href="#">ClickLabel</a>        | "ClickLabel" event handler (virtual)                        |

## Create

Creates new CCheckBox control.

```
virtual bool Create(  
    const long    chart,      // chart ID  
    const string  name,      // name  
    const int     subwin,     // chart subwindow  
    const int     x1,        // x1 coordinate  
    const int     y1,        // y1 coordinate  
    const int     x2,        // x2 coordinate  
    const int     y2         // y2 coordinate  
)
```

### Parameters

*chart*

[in] chart ID.

*name*

[in] Unique name of the control.

*subwin*

[in] Chart subwindow.

*x1*

[in] X coordinate of the upper left corner.

*y1*

[in] Y coordinate of the upper left corner.

*x2*

[in] X coordinate of the lower right corner.

*y2*

[in] Y coordinate of the lower right corner.

### Returned value

true if successful, otherwise false.

## OnEvent

Chart event handler.

```
virtual bool OnEvent(  
    const int      id,           // ID  
    const long&    lparam,      // event parameter of long type  
    const double&  dparam,      // event parameter of double type  
    const string&  sparam       // event parameter of string type  
)
```

### Parameters

*id*

[in] Event ID.

*lparam*

[in] Event parameter of [long](#) type, passed by reference.

*dparam*

[in] Event parameter of [double](#) type, passed by reference.

*sparam*

[in] Event parameter of [string](#) type, passed by reference.

### Returned value

true - if the event has been processed, otherwise false.

## Text (Get method)

Gets text of the label, associated with the control.

```
string Text()
```

### Returned value

Text of the label.

## Text (Set method)

Sets text of the label, associated with the control.

```
bool Text (  
    const string value    // text  
)
```

### Parameters

*value*

[in] New text of the label.

### Returned value

true if successful, otherwise false.

## Color (Get method)

Gets color of the label, associated with the control.

```
color Color() const
```

### Returned value

Label color.

## Color (Set method)

Sets color of the label, associated with the control.

```
bool Color(  
    const color value    // color  
)
```

### Parameters

*value*

[in] New label color.

### Returned value

true if successful, otherwise false.

## Checked (Get method)

Gets state of the control.

```
bool Checked() const
```

### Returned value

State of the control.

## Checked (Set method)

Sets state of the control.

```
bool Checked(  
    const bool flag    // state  
)
```

### Parameters

*flag*

[in] New state.

### Returned value

true if successful, otherwise false.

## Value (Get method)

Gets the value, associated with the control.

```
int Value() const
```

### Returned value

The value, associated with the control.

## Value (Set method)

Sets the value, associated with the control.

```
void Value(  
    const int value    // new value  
)
```

### Parameters

*value*

[in] New value.

### Returned value

None.



## CreateButton

Creates dependent control (button).

```
virtual bool CreateButton()
```

### Returned value

true if successful, otherwise false.

## CreateLabel

Creates dependent control (label).

```
virtual bool CreateLabel()
```

### Returned value

true if successful, otherwise false.

## OnClickButton

The control "ClickButton" (mouse click on the button) event handler.

```
virtual bool OnClickButton()
```

### Returned value

true if the event has been processed, otherwise false.

## OnClickLabel

The control "ClickLabel" (mouse click on the label) event handler.

```
virtual bool OnClickLabel ()
```

### Returned value

true if the event has been processed, otherwise false.

## CCheckGroup

CCheckGroup is a class of the CheckGroup complex control (with dependent controls).

### Description

CCheckGroup provides the possibility for creation of controls, which allow to display and edit flags.

### Declaration

```
class CCheckGroup : public CWndClient
```

### Title

```
#include <Controls\CheckGroup.mqh>
```

### Class Methods

| Create                            |                                                                       |
|-----------------------------------|-----------------------------------------------------------------------|
| <a href="#">Create</a>            | Creates control                                                       |
| Chart event handlers              |                                                                       |
| <a href="#">OnEvent</a>           | Event handler for all chart events                                    |
| Add                               |                                                                       |
| <a href="#">AddItem</a>           | Adds new item                                                         |
| Read-only data                    |                                                                       |
| <a href="#">Value</a>             | Gets the value, associated with the control                           |
| Dependent controls                |                                                                       |
| <a href="#">CreateButton</a>      | Creates new CCheckBox item                                            |
| Dependent controls event handlers |                                                                       |
| <a href="#">OnVScrollShow</a>     | "Show" event handler (virtual) of VScroll dependent control           |
| <a href="#">OnVScrollHide</a>     | "Hide" event handler (virtual) of VScroll dependent control           |
| <a href="#">OnScrollLineDown</a>  | "ScrollLineUp" event handler (virtual) of VScroll dependent control   |
| <a href="#">OnScrollLineUp</a>    | "ScrollLineDown" event handler (virtual) of VScroll dependent control |
| <a href="#">OnChangeItem</a>      | "ChangeItem" event handler (virtual) of VScroll dependent control     |
| Redraw                            |                                                                       |
| <a href="#">Redraw</a>            | Redraws the group                                                     |

RowState

Sets the state of the specified item

## Create

Creates new CCheckGroup control.

```
virtual bool Create(  
    const long    chart,      // chart ID  
    const string  name,      // name  
    const int     subwin,     // chart subwindow  
    const int     x1,        // x1 coordinate  
    const int     y1,        // y1 coordinate  
    const int     x2,        // x2 coordinate  
    const int     y2         // y2 coordinate  
)
```

### Parameters

*chart*

[in] chart ID.

*name*

[in] Unique name of the control.

*subwin*

[in] Chart subwindow.

*x1*

[in] X coordinate of the upper left corner.

*y1*

[in] Y coordinate of the upper left corner.

*x2*

[in] X coordinate of the lower right corner.

*y2*

[in] Y coordinate of the lower right corner.

### Returned value

true if successful, otherwise false.

## OnEvent

Chart event handler.

```
virtual bool OnEvent(  
    const int      id,           // ID  
    const long&    lparam,      // event parameter of long type  
    const double&  dparam,      // event parameter of double type  
    const string&  sparam       // event parameter of string type  
)
```

### Parameters

*id*

[in] Event ID.

*lparam*

[in] Event parameter of [long](#) type, passed by reference.

*dparam*

[in] Event parameter of [double](#) type, passed by reference.

*sparam*

[in] Event parameter of [string](#) type, passed by reference.

### Returned value

true - if the event has been processed, otherwise false.



## AddItem

Adds an item to the control.

```
bool AddItem(  
    const string item,      // text  
    const long  value      // value  
)
```

### Parameters

*item*

[in] Text.

*value=0*

[in] Value of [long](#) type.

### Returned value

true if successful, otherwise false.

## Value

Gets the value, associated with the control.

```
long Value()
```

### Returned value

The value, associated with the control.

### Note

The value depends on state of all items of the CCheckGroup.

## CreateButton

Creates new CCheckBox class instance on the specified index.

```
bool CreateButton(  
    int index    // index  
)
```

### Parameters

*index*

[in] Index of the new item in the CCheckGroup.

### Returned value

true if successful, otherwise false.

## OnVScrollShow

The control "VScrollShow" (vertical scroll bar show) event handler.

```
virtual bool OnVScrollShow()
```

### Returned value

true if the event has been processed, otherwise false.

## OnVScrollHide

The control "VScrollHide" (vertical scroll bar hide) event handler.

```
virtual bool OnVScrollHide()
```

### Returned value

true if the event has been processed, otherwise false.

## OnScrollLineDown

The control "ScrollLineDown" (vertical scroll line down) event handler.

```
virtual bool OnScrollLineDown ()
```

### Returned value

true if the event has been processed, otherwise false.

## OnScrollLineUp

The control "ScrollLineUp" (vertical scroll line up) event handler.

```
virtual bool OnScrollLineUp()
```

### Returned value

true if the event has been processed, otherwise false.

## OnChangeItem

The control "ChangeItem" (item change) event handler.

```
virtual bool OnChangeItem(  
    const int index    // index  
)
```

### Parameters

*index*

[in] Index of the changed item.

### Returned value

true if the event has been processed, otherwise false.



## Redraw

Redraws the control.

```
bool Redraw()
```

### Returned value

true if successful, otherwise false.

## RowState

Sets the state of the specified item.

```
bool RowState(  
    const int   index,      // item index  
    const bool  select      // state  
)
```

### Parameters

*index*

[in] Item index to change.

*select*

[in] New state.

### Returned value

true if successful, otherwise false.

# CRadioButton

CRadioButton is a class of RadioButton complex control.

## Description

CRadioButton itself is not used, it used for creation of [CRadioGroup](#) items.

## Declaration

```
class CRadioButton : public CWndContainer
```

## Title

```
#include <Controls\RadioButton.mqh>
```

## Class Methods

| Create                            |                                                                |
|-----------------------------------|----------------------------------------------------------------|
| <a href="#">Create</a>            | Creates control                                                |
| Chart event handlers              |                                                                |
| <a href="#">OnEvent</a>           | Event handler for all chart events                             |
| Properties                        |                                                                |
| <a href="#">Text</a>              | Gets/Sets the text label, associated with the control          |
| <a href="#">Color</a>             | Gets/Sets the color of text label, associated with the control |
| State                             |                                                                |
| <a href="#">State</a>             | Gets/Sets the state                                            |
| Dependent controls                |                                                                |
| <a href="#">CreateButton</a>      | Creates button                                                 |
| <a href="#">CreateLabel</a>       | Creates label                                                  |
| Dependent controls event handlers |                                                                |
| <a href="#">OnClickButton</a>     | "ClickButton" event handler (virtual)                          |
| <a href="#">OnClickLabel</a>      | "ClickLabel" event handler (virtual)                           |

## Create

Creates new CRadioButton control.

```
virtual bool Create(  
    const long    chart,      // chart ID  
    const string  name,      // name  
    const int     subwin,     // chart subwindow  
    const int     x1,        // x1 coordinate  
    const int     y1,        // y1 coordinate  
    const int     x2,        // x2 coordinate  
    const int     y2         // y2 coordinate  
)
```

### Parameters

*chart*

[in] chart ID.

*name*

[in] Unique name of the control.

*subwin*

[in] Chart subwindow.

*x1*

[in] X coordinate of the upper left corner.

*y1*

[in] Y coordinate of the upper left corner.

*x2*

[in] X coordinate of the lower right corner.

*y2*

[in] Y coordinate of the lower right corner.

### Returned value

true if successful, otherwise false.

## OnEvent

Chart event handler.

```
virtual bool OnEvent(  
    const int      id,           // ID  
    const long&    lparam,       // event parameter of long type  
    const double&  dparam,       // event parameter of double type  
    const string&  sparam        // event parameter of string type  
)
```

### Parameters

*id*

[in] Event ID.

*lparam*

[in] Event parameter of [long](#) type, passed by reference.

*dparam*

[in] Event parameter of [double](#) type, passed by reference.

*sparam*

[in] Event parameter of [string](#) type, passed by reference.

### Returned value

true - if the event has been processed, otherwise false.

## Text (Get method)

Gets text of the label, associated with the control.

```
string Text()
```

### Returned value

Text of the label.

## Text (Set method)

Sets text of the label, associated with the control.

```
bool Text(  
    const string value    // text  
)
```

### Parameters

*value*

[in] New text of the label.

### Returned value

true if successful, otherwise false.

## Color (Get method)

Gets color of the label, associated with the control.

```
color Color() const
```

### Returned value

Label color.

## Color (Set method)

Sets color of the label, associated with the control.

```
bool Color(  
    const color value    // color  
)
```

### Parameters

*value*

[in] New label color.

### Returned value

true if successful, otherwise false.

## State (Get method)

Gets the button state.

```
bool State() const
```

### Returned value

Button state.

## State (Set method)

Sets the button state.

```
bool State(  
    const bool flag    // flag  
)
```

### Parameters

*flag*

[in] New button state.

### Returned value

true if successful, otherwise false.



## CreateButton

Creates button.

```
virtual bool CreateButton()
```

### Returned value

true if successful, otherwise false.

## CreateLabel

Creates label.

```
virtual bool CreateLabel()
```

### Returned value

true if successful, otherwise false.

## OnClickButton

The control "ClickButton" (mouse click) event handler.

```
virtual bool OnClickButton()
```

### Returned value

true if the event has been processed, otherwise false.

## OnClickLabel

The control "ClickLabel" (mouse click on the label) event handler.

```
virtual bool OnClickLabel ()
```

### Returned value

true if the event has been processed, otherwise false.

## CRadioGroup

CRadioGroup is class of the RadioGroup complex control (with dependent controls).

### Description

CRadioGroup enables the user to select a single option from a group of choices when paired with other [CRadioButton](#) controls.

### Declaration

```
class CRadioGroup : public CWndClient
```

### Title

```
#include <Controls\RadioGroup.mqh>
```

### Class Methods

| Create                            |                                                                       |
|-----------------------------------|-----------------------------------------------------------------------|
| <a href="#">Create</a>            | Creates control                                                       |
| Chart event handlers              |                                                                       |
| <a href="#">OnEvent</a>           | Event handler of all chart events                                     |
| Add                               |                                                                       |
| <a href="#">AddItem</a>           | Adds new item                                                         |
| Read-only data                    |                                                                       |
| <a href="#">Value</a>             | Gets the value, associated with the control                           |
| Dependent controls                |                                                                       |
| <a href="#">CreateButton</a>      | Creates new CRadioButton item                                         |
| Dependent controls event handlers |                                                                       |
| <a href="#">OnVScrollShow</a>     | "Show" event handler (virtual) of VScroll dependent control           |
| <a href="#">OnVScrollHide</a>     | "Hide" event handler (virtual) of VScroll dependent control           |
| <a href="#">OnScrollLineDown</a>  | "ScrollLineDown" event handler (virtual) of VScroll dependent control |
| <a href="#">OnScrollLineUp</a>    | "ScrollLineUp" event handler (virtual) of VScroll dependent control   |
| <a href="#">OnChangeItem</a>      | "ChangeItem" event handler (virtual)                                  |
| Redraw                            |                                                                       |
| <a href="#">Redraw</a>            | Redraws the group items                                               |

|                 |                                      |
|-----------------|--------------------------------------|
| <u>RowState</u> | Sets the state of the specified item |
| <u>Select</u>   | Selects current item                 |

## Create

Creates new CRadioGroup control.

```
virtual bool Create(  
    const long    chart,      // chart ID  
    const string  name,      // name  
    const int     subwin,     // chart subwindow  
    const int     x1,        // x1 coordinate  
    const int     y1,        // y1 coordinate  
    const int     x2,        // x2 coordinate  
    const int     y2         // y2 coordinate  
)
```

### Parameters

*chart*

[in] chart ID.

*name*

[in] Unique name of the control.

*subwin*

[in] Chart subwindow.

*x1*

[in] X coordinate of the upper left corner.

*y1*

[in] Y coordinate of the upper left corner.

*x2*

[in] X coordinate of the lower right corner.

*y2*

[in] Y coordinate of the lower right corner.

### Returned value

true if successful, otherwise false.

## OnEvent

Chart event handler.

```
virtual bool OnEvent(  
    const int      id,           // ID  
    const long&    lparam,       // event parameter of long type  
    const double&  dparam,       // event parameter of double type  
    const string&  sparam        // event parameter of string type  
)
```

### Parameters

*id*

[in] Event ID.

*lparam*

[in] Event parameter of [long](#) type, passed by reference.

*dparam*

[in] Event parameter of [double](#) type, passed by reference.

*sparam*

[in] Event parameter of [string](#) type, passed by reference.

### Returned value

true - if the event has been processed, otherwise false.



## AddItem

Adds an item to the control.

```
bool AddItem(  
    const string item,      // text  
    const long  value=0    // value  
)
```

### Parameters

*item*

[in] Text.

*value=0*

[in] Value of [long](#) type.

### Returned value

true if successful, otherwise false.

## Value

Gets the value, associated with the control.

```
long Value()
```

### Returned value

The value, associated with the control.

### Note

The value depends on state of all CRadioButton items of the CRadioGroup control.

## CreateButton

Creates new CRadioButton class instance on the specified index.

```
bool CreateButton(  
    const int index    // index  
)
```

### Parameters

*index*

[in] Index of the new item in the CRadioGroup.

### Returned value

true if successful, otherwise false.

## OnVScrollShow

The control "VScrollShow" (vertical scroll bar show) event handler.

```
virtual bool OnVScrollShow()
```

### Returned value

true if the event has been processed, otherwise false.

## OnVScrollHide

The control "VScrollHide" (vertical scroll bar hide) event handler.

```
virtual bool OnVScrollHide()
```

### Returned value

true if the event has been processed, otherwise false.

## OnScrollLineDown

The control "ScrollLineDown" (vertical scroll line down) event handler.

```
virtual bool OnScrollLineDown ()
```

### Returned value

true if the event has been processed, otherwise false.

## OnScrollLineUp

The control "ScrollLineUp" (vertical scroll line up) event handler.

```
virtual bool OnScrollLineUp()
```

### Returned value

true if the event has been processed, otherwise false.

## OnChangeItem

The control "ChangeItem" (item change) event handler.

```
virtual bool OnChangeItem(  
    const int index    // index  
)
```

### Parameters

*index*

[in] Index of the changed item.

### Returned value

true if the event has been processed, otherwise false.



## Redraw

Redraws the control.

```
bool Redraw()
```

### Returned value

true if successful, otherwise false.

## RowState

Sets the state of the specified item.

```
bool RowState(  
    const int   index,      // item index  
    const bool  select      // state  
)
```

### Parameters

*index*

[in] Item index to change.

*select*

[in] New state.

### Returned value

true if successful, otherwise false.

## Select

Selects current item.

```
void Select(  
    const int index    // index  
)
```

### Parameters

*index*

[in] Item index to select.

### Returned value

None.

## CSpinEdit

CSpinEdit is a class of SpinEdit complex control (with dependent controls).

### Description

CSpinEdit class is intended for creation of controls, which allows to edit the values of integer type. It automatically increases data when pressing the upper button, but decreases if pressing the lower button.

### Declaration

```
class CSpinEdit : public CWndContainer
```

### Title

```
#include <Controls\SpinEdit.mqh>
```

### Class Methods

| Create                            |                                              |
|-----------------------------------|----------------------------------------------|
| <a href="#">Create</a>            | Creates control                              |
| Chart event handlers              |                                              |
| <a href="#">OnEvent</a>           | Event handler of all chart events            |
| Properties                        |                                              |
| <a href="#">MinValue</a>          | Gets/Sets the minimal allowed value          |
| <a href="#">MaxValue</a>          | Gets/Sets the maximal allowed value          |
| State                             |                                              |
| <a href="#">Value</a>             | Gets/Sets the current value                  |
| Dependent controls                |                                              |
| <a href="#">CreateEdit</a>        | Creates dependent control (edit)             |
| <a href="#">CreateInc</a>         | Creates dependent control (increment button) |
| <a href="#">CreateDec</a>         | Creates dependent control (decrement button) |
| Dependent controls event handlers |                                              |
| <a href="#">OnClickInc</a>        | "ClickInc" event handler (virtual)           |
| <a href="#">OnClickDec</a>        | "ClickDec" event handler (virtual)           |
| Internal event handlers           |                                              |
| <a href="#">OnChangeValue</a>     | "ChangeValue" event handler (virtual)        |

## Create

Creates new CSpinEdit control.

```
virtual bool Create(  
    const long    chart,      // chart ID  
    const string  name,      // name  
    const int     subwin,     // chart subwindow  
    const int     x1,        // x1 coordinate  
    const int     y1,        // y1 coordinate  
    const int     x2,        // x2 coordinate  
    const int     y2         // y2 coordinate  
)
```

### Parameters

*chart*

[in] chart ID.

*name*

[in] Unique name of the control.

*subwin*

[in] Chart subwindow.

*x1*

[in] X coordinate of the upper left corner.

*y1*

[in] Y coordinate of the upper left corner.

*x2*

[in] X coordinate of the lower right corner.

*y2*

[in] Y coordinate of the lower right corner.

### Returned value

true if successful, otherwise false.

## OnEvent

Chart event handler.

```
virtual bool OnEvent(  
    const int      id,           // ID  
    const long&    lparam,       // event parameter of long type  
    const double&  dparam,       // event parameter of double type  
    const string&  sparam        // event parameter of string type  
)
```

### Parameters

*id*

[in] Event ID.

*lparam*

[in] Event parameter of [long](#) type, passed by reference.

*dparam*

[in] Event parameter of [double](#) type, passed by reference.

*sparam*

[in] Event parameter of [string](#) type, passed by reference.

### Returned value

true - if the event has been processed, otherwise false.

## MinValue (Get method)

Gets the value of "MinValue" property (minimal allowed value) of the control.

```
int MinValue() const
```

### Returned value

The value of "MinValue" property.

## MinValue (Set method)

Sets the value of "MinValue" property (minimal allowed value) of the control.

```
void MinValue(  
    const int value    // new value  
)
```

### Parameters

*value*

[in] New value of "MinValue" property.

### Returned value

None.

## MaxValue (Get method)

Gets the value of "MaxValue" property (maximal allowed value) of the control.

```
int MaxValue() const
```

### Returned value

The value of "MaxValue" property.

## MaxValue (Set method)

Sets the value of "MaxValue" property (maximal allowed value) of the control.

```
void MaxValue(  
    const int value    // new value  
)
```

### Parameters

*value*

[in] New value of "MaxValue" property.

### Returned value

None.



## Value (Get method)

Gets the "Value" property (current value) of the control.

```
int Value() const
```

### Returned value

The "Value" property.

## Value (Set method)

Sets the "Value" property (current value) of the control.

```
void Value(  
    const int value    // value  
)
```

### Parameters

*value*

[in] New "Value" property.

### Returned value

None.

## CreateEdit

Creates dependent control (CEdit).

```
virtual bool CreateEdit()
```

### Returned value

true if successful, otherwise false.

## CreateInc

Creates dependent control (increment button).

```
virtual bool CreateInc()
```

### Returned value

true if successful, otherwise false.

## CreateDec

Creates dependent control (decrement button).

```
virtual bool CreateDec()
```

### Returned value

true if successful, otherwise false.

## OnClickInc

The control "ClickInc" (mouse click on the increment button) event handler.

```
virtual bool OnClickInc()
```

### Returned value

true if the event has been processed, otherwise false.

## OnClickDec

The control "ClickDec" (mouse click on the decrement button) event handler.

```
virtual bool OnClickDec()
```

### Returned value

true if the event has been processed, otherwise false.

## OnChangeValue

The control "ChangeValue" event handler.

```
virtual bool OnChangeValue ()
```

### Returned value

true if the event has been processed, otherwise false.

## CDialog

CDialog is class of the Dialog complex control.

### Description

CDialog class is intended to combine the controls with different functions in the group.

### Declaration

```
class CDialog : public CWndContainer
```

### Title

```
#include <Controls\Dialog.mqh>
```

### Class Methods

| Create                             |                                                                       |
|------------------------------------|-----------------------------------------------------------------------|
| <a href="#">Create</a>             | Creates control                                                       |
| Chart event handlers               |                                                                       |
| <a href="#">OnEvent</a>            | Event handler of all chart events                                     |
| Properties                         |                                                                       |
| <a href="#">Caption</a>            | Gets/Sets the value of the "Caption" property                         |
| Add                                |                                                                       |
| <a href="#">Add</a>                | Adds control to the client area                                       |
| Dependent controls                 |                                                                       |
| <a href="#">CreateWhiteBorder</a>  | Creates dependent control (white border)                              |
| <a href="#">CreateBackground</a>   | Creates dependent control (background)                                |
| <a href="#">CreateCaption</a>      | Creates dependent control (caption)                                   |
| <a href="#">CreateButtonClose</a>  | Creates dependent control (close button)                              |
| <a href="#">CreateClientArea</a>   | Creates dependent control (client area)                               |
| Dependent controls event handlers  |                                                                       |
| <a href="#">OnClickCaption</a>     | "ClickCaption" event handler                                          |
| <a href="#">OnClickButtonClose</a> | "ClickButtonClose" event handler                                      |
| Access to client area              |                                                                       |
| <a href="#">ClientAreaVisible</a>  | Sets a value indicating whether the client area is visible            |
| <a href="#">ClientAreaLeft</a>     | Gets X coordinate of the upper-left corner of the control client area |



|                                            |                                                                        |
|--------------------------------------------|------------------------------------------------------------------------|
| <a href="#"><u>ClientAreaTop</u></a>       | Gets Y coordinate of the upper-left corner of the control client area  |
| <a href="#"><u>ClientAreaRight</u></a>     | Gets X coordinate of the lower-right corner of the control client area |
| <a href="#"><u>ClientAreaBottom</u></a>    | Gets Y coordinate of the lower-right corner of the control client area |
| <a href="#"><u>ClientAreaWidth</u></a>     | Gets the client area width                                             |
| <a href="#"><u>ClientAreaHeight</u></a>    | Gets the client area height                                            |
| <b>Drag event handlers</b>                 |                                                                        |
| <a href="#"><u>OnDialogDragStart</u></a>   | "DialogDragStart" event handler (virtual)                              |
| <a href="#"><u>OnDialogDragProcess</u></a> | "DialogDragProcess" event handler (virtual)                            |
| <a href="#"><u>OnDialogDragEnd</u></a>     | "DialogDragEnd" event handler (virtual)                                |

## Create

Creates new CDialog control.

```
virtual bool Create(  
    const long    chart,      // chart ID  
    const string  name,      // name  
    const int     subwin,    // chart subwindow  
    const int     x1,        // x1 coordinate  
    const int     y1,        // y1 coordinate  
    const int     x2,        // x2 coordinate  
    const int     y2        // y2 coordinate  
)
```

### Parameters

*chart*

[in] chart ID.

*name*

[in] Unique name of the control.

*subwin*

[in] Chart subwindow.

*x1*

[in] X coordinate of the upper left corner.

*y1*

[in] Y coordinate of the upper left corner.

*x2*

[in] X coordinate of the lower right corner.

*y2*

[in] Y coordinate of the lower right corner.

### Returned value

true if successful, otherwise false.

## OnEvent

Chart event handler.

```
virtual bool OnEvent(  
    const int      id,           // ID  
    const long&    lparam,       // event parameter of long type  
    const double&  dparam,       // event parameter of double type  
    const string&  sparam        // event parameter of string type  
)
```

### Parameters

*id*

[in] Event ID.

*lparam*

[in] Event parameter of [long](#) type, passed by reference.

*dparam*

[in] Event parameter of [double](#) type, passed by reference.

*sparam*

[in] Event parameter of [string](#) type, passed by reference.

### Returned value

true - if the event has been processed, otherwise false.

## Caption (Get method)

Gets the "Caption" property of the CDialog control.

```
string MinValue() const
```

### Returned value

The "Caption" property.

## Caption (Set method)

Sets the "Caption" property of the CDialog control.

```
bool Caption(  
    const string text    // text  
)
```

### Parameters

*text*

[in] New value of "Caption" property.

### Returned value

true if successful, otherwise false.

## Add

Adds control to the client area by pointer.

```
bool Add(  
    CWnd *control,           // pointer  
)
```

### Parameters

*control*

[in] Pointer to control.

### Returned value

true if successful, otherwise false.

## Add

Adds control to the client area by reference.

```
bool Add(  
    CWnd &control,           // reference  
)
```

### Parameters

*control*

[in] Reference to control.

### Returned value

true if successful, otherwise false.

## CreateWhiteBorder

Creates dependent control (white border).

```
virtual bool CreateWhiteBorder()
```

### Returned value

true if successful, otherwise false.

## CreateBackground

Creates dependent control (background).

```
virtual bool CreateBackground()
```

### Returned value

true if successful, otherwise false.

## CreateCaption

Creates dependent control (caption).

```
virtual bool CreateCaption()
```

### Returned value

true if successful, otherwise false.



## CreateButtonClose

Creates dependent control (close button)

```
virtual bool CreateButtonClose()
```

### Returned value

true if successful, otherwise false.

## CreateClientArea

Creates dependent control (client area).

```
virtual bool CreateClientArea ()
```

### Returned value

true if successful, otherwise false.

## OnClickCaption

The control "ClickCaption" event handler.

```
virtual bool OnClickCaption()
```

### Returned value

true if successful, otherwise false.

## OnClickButtonClose

The control "ClickButtonClose" event handler.

```
virtual bool OnClickButtonClose()
```

### Returned value

true if successful, otherwise false.

## ClientAreaVisible

Sets a flag indicating whether the client area is visible.

```
bool ClientAreaVisible(  
    const bool visible    // visibility flag  
)
```

### Parameters

*visible*

[in] Visibility flag.

### Returned value

true if successful, otherwise false.

## ClientAreaLeft

Gets X coordinate of the upper-left corner of the control client area.

```
int ClientAreaLeft()
```

### Returned value

The X coordinate of the upper-left corner of the control client area.

## ClientAreaTop

Gets Y coordinate of the upper-left corner of the control client area.

```
int ClientAreaTop()
```

### Returned value

The Y coordinate of the upper-left corner of the control client area.

## ClientAreaRight

Gets X coordinate of the lower-right corner of the control client area.

```
int ClientAreaTop()
```

### Returned value

The X coordinate of the lower-right corner of the control client area.



## ClientAreaBottom

Gets Y coordinate of the lower-right corner of the control client area.

```
int ClientAreaBottom()
```

### Returned value

The Y coordinate of the lower-right corner of the control client area.

## ClientAreaWidth

Gets the width of the control client area.

```
int ClientAreaWidth()
```

### Returned value

The width of the client area.

## ClientAreaHeight

Gets the height of the control client area.

```
int ClientAreaHeight ()
```

### Returned value

The height of the control client area.

## OnDialogDragStart

The control "DialogDragStart" event handler.

```
virtual bool OnDialogDragStart ()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The "DialogDragStart" event occurs at start of the drag of the control.

## OnDialogDragProcess

The control "DialogDragProcess" event handler.

```
virtual bool OnDialogDragProcess()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The "DialogDragProcess" event occurs when the control is dragged.

## OnDialogDragEnd

The control "DialogDragEnd" event handler.

```
virtual bool OnDialogDragEnd()
```

### Returned value

true if the event has been processed, otherwise false.

### Note

The "DialogDragEnd" event occurs at the end of the drag of the control.

## CAppDialog

CAppDialog is a class of Application Dialog complex control (with dependent controls).

### Description

CAppDialog class is intended to combine the controls with different functions in the group inside the MQL5 program.

### Declaration

```
class CAppDialog : public CDialog
```

### Title

```
#include <Controls\Dialog.mqh>
```

### Class Methods

| Create and destroy           |                                                              |
|------------------------------|--------------------------------------------------------------|
| <a href="#">Create</a>       | Creates control                                              |
| <a href="#">Destroy</a>      | Destroys control                                             |
| Events processing            |                                                              |
| <a href="#">OnEvent</a>      | Event handler of all chart events                            |
| Run                          |                                                              |
| <a href="#">Run</a>          | Runs control                                                 |
| Chart events processing      |                                                              |
| <a href="#">ChartEvent</a>   | Event handler of all chart events                            |
| Settings                     |                                                              |
| <a href="#">Minimized</a>    | Sets a value indicating whether the control is minimized     |
| Save/Load                    |                                                              |
| <a href="#">IniFileSave</a>  | Saves the control state to file                              |
| <a href="#">IniFileLoad</a>  | Loads the control state from file                            |
| <a href="#">IniFileName</a>  | Sets the file name for loading/saving the control state      |
| <a href="#">IniFileExt</a>   | Sets the file extension for loading/saving the control state |
| Initialization               |                                                              |
| <a href="#">CreateCommon</a> | Common initialization method                                 |

|                                                  |                                                                   |
|--------------------------------------------------|-------------------------------------------------------------------|
| <a href="#"><u>CreateExpert</u></a>              | Initialization method for working in Expert Advisors              |
| <a href="#"><u>CreateIndicator</u></a>           | Initialization method for working in indicators                   |
| <b>Dependent controls</b>                        |                                                                   |
| <a href="#"><u>CreateButtonMinMax</u></a>        | Creates dependent controls (minimize/maximize buttons)            |
| <b>Dependent controls event handlers</b>         |                                                                   |
| <a href="#"><u>OnClickButtonClose</u></a>        | "ClickButtonClose" event handler (virtual)                        |
| <a href="#"><u>OnClickButtonMinMax</u></a>       | "ClickButtonMinMax" event handler (virtual)                       |
| <b>External events</b>                           |                                                                   |
| <a href="#"><u>OnAnotherApplicationClose</u></a> | Event handler of external events (virtual)                        |
| <b>Methods</b>                                   |                                                                   |
| <a href="#"><u>Rebound</u></a>                   | Sets new coordinates of the control using CRect class coordinates |
| <a href="#"><u>Minimize</u></a>                  | Shows the control in the minimized state                          |
| <a href="#"><u>Maximize</u></a>                  | Shows the control in the maximized (restored) state               |
| <a href="#"><u>CreateInstanceld</u></a>          | Creates an unique Id for the names of the control objects         |
| <a href="#"><u>ProgramName</u></a>               | Gets the name of MQL5 program                                     |
| <a href="#"><u>SubwinOff</u></a>                 | Get the Y offset of the control subwindow                         |



## Create

Creates new CAppDialog control.

```
virtual bool Create(  
    const long    chart,      // chart ID  
    const string  name,      // name  
    const int     subwin,     // chart subwindow  
    const int     x1,        // x1 coordinate  
    const int     y1,        // y1 coordinate  
    const int     x2,        // x2 coordinate  
    const int     y2         // y2 coordinate  
)
```

### Parameters

*chart*

[in] chart ID.

*name*

[in] Unique name of the control.

*subwin*

[in] Chart subwindow.

*x1*

[in] X coordinate of the upper left corner.

*y1*

[in] Y coordinate of the upper left corner.

*x2*

[in] X coordinate of the lower right corner.

*y2*

[in] Y coordinate of the lower right corner.

### Returned value

true if successful, otherwise false.

## Destroy

Destroys control.

```
virtual bool Destroy()
```

### Returned value

true if successful, otherwise false.

## OnEvent

Chart event handler.

```
virtual bool OnEvent(  
    const int      id,           // ID  
    const long&    lparam,       // event parameter of long type  
    const double&  dparam,       // event parameter of double type  
    const string&  sparam        // event parameter of string type  
)
```

### Parameters

*id*

[in] Event ID.

*lparam*

[in] Event parameter of [long](#) type, passed by reference.

*dparam*

[in] Event parameter of [double](#) type, passed by reference.

*sparam*

[in] Event parameter of [string](#) type, passed by reference.

### Returned value

true - if the event has been processed, otherwise false.

## Run

Runs control.

```
bool Run()
```

### Returned value

true if successful, otherwise false.

## ChartEvent

Chart event handler.

```
virtual bool ChartEvent(  
    const int      id,           // ID  
    const long&    lparam,       // event parameter of long type  
    const double&  dparam,       // event parameter of double type  
    const string&  sparam        // event parameter of string type  
)
```

### Parameters

*id*

[in] Event ID.

*lparam*

[in] Event parameter of [long](#) type, passed by reference.

*dparam*

[in] Event parameter of [double](#) type, passed by reference.

*sparam*

[in] Event parameter of [string](#) type, passed by reference.

### Returned value

true - if the event has been processed, otherwise false.

## Minimized

Sets the value of "Minimized" property of the control.

```
bool Minimized(  
    const bool flag    // state  
)
```

### Parameters

*flag*

[in] New state.

### Returned value

true if successful, otherwise false.

## IniFileSave

Saves the control state to file.

```
void IniFileSave()
```

### Returned value

true if successful, otherwise false.

## IniFileLoad

Loads the control state from file.

```
void IniFileLoad()
```

### Returned value

true if successful, otherwise false.



## IniFileName

Sets the file name for loading/saving the control state.

```
virtual string IniFileName() const
```

### Returned value

File name for loading/saving of the control state.

### Note

The file name include the name of the Expert Advisor/indicator and working symbol, on which MQL5 program is launched.

## IniFileExt

Sets the file extension for loading/saving the control state.

```
virtual string IniFileExt() const
```

### Returned value

File extension, used for loading/saving of the control state.

## CreateCommon

Common initialization method.

```
bool CreateCommon(  
    const long    chart,      // chart ID  
    const string  name,      // name  
    const int     subwin,     // chart subwindow  
)
```

### Parameters

*chart*

[in] chart ID.

*name*

[in] Unique name of the control.

*subwin*

[in] Chart subwindow.

### Returned value

true if successful, otherwise false.

## CreateExpert

Initialization method for working in Expert Advisors.

```
bool CreateExpert (
    const int    x1,          // x1 coordinate
    const int    y1,          // y1 coordinate
    const int    x2,          // x2 coordinate
    const int    y2           // y2 coordinate
)
```

### Parameters

*x1*

[in] X coordinate of the upper left corner.

*y1*

[in] Y coordinate of the upper left corner.

*x2*

[in] X coordinate of the lower right corner.

*y2*

[in] Y coordinate of the lower right corner.

### Returned value

true if successful, otherwise false.

## CreateIndicator

Initialization method for working in indicators.

```
bool CreateIndicator(  
    const int    x1,          // x1 coordinate  
    const int    y1,          // y1 coordinate  
    const int    x2,          // x2 coordinate  
    const int    y2           // y2 coordinate  
)
```

### Parameters

*x1*

[in] X coordinate of the upper left corner.

*y1*

[in] Y coordinate of the upper left corner.

*x2*

[in] X coordinate of the lower right corner.

*y2*

[in] Y coordinate of the lower right corner.

### Returned value

true if successful, otherwise false.

## CreateButtonMinMax

Creates dependent controls (minimize/maximize buttons).

```
virtual void CreateButtonMinMax()
```

### Returned value

None.

## OnClickButtonClose

The control "ClickButtonClose" (mouse click on close button) event handler.

```
virtual void OnClickButtonClose()
```

### Returned value

None.

## OnClickButtonMinMax

The control "ClickButtonMinMax" (mouse click on minimize/maximize button) event handler.

```
virtual void OnClickButtonClose()
```

### Returned value

None.



## OnAnotherApplicationClose

Event handler of external events.

```
virtual void OnAnotherApplicationClose()
```

### Returned value

None.

## Rebound

Sets new coordinates of the control using CRect class coordinates.

```
bool Rebound(  
    const & CRect rect    // CRect class  
)
```

### Returned value

true if successful, otherwise false.

## Minimize

Shows the control in the minimized state.

```
virtual void Minimize()
```

### Returned value

true if successful, otherwise false.

## Maximize

Shows the control in the maximized (restored) state.

```
virtual void Maximize()
```

### Returned value

true if successful, otherwise false.

## CreateInstanceId

Creates an unique Id for the names of the control objects.

```
string CreateInstanceId()
```

### Returned value

Prefix for the object names.

## ProgramName

Gets the name of the MQL5 program.

```
string ProgramName ()
```

### Returned value

Name of the MQL5 program.

## SubwinOff

Get the Y offset of the control subwindow.

```
void SubwinOff()
```

### Returned value

None.

## Übergang von MQL4

Die Sprache MQL5 ist Entwicklung ihres Vorgängers - der Sprache MQL4, in der viele Indikatoren, Scripts und Experten geschrieben werden. Obwohl die neue Programmiersprache mit der Sprache der früheren Generation maximal kompatibel ist. Es gibt es eine Reihe der Unterschiede zwischen diesen Sprachen. Bei der Portierung der Programme muss man diese Unterschiede kennen.

In diesem Abschnitt gibt es die notwendige Information, um die Adaptierung der Codes zur neuen Sprache MQL5 für Programmierer, die MQL4 gut kennen, zu ermöglichen.

Vor allem muss bemerkt werden:

- Funktionen `start()`, `init()` und `deinit()` fehlen;
- Anzahl der Indikatorpuffer ist nicht begrenzt;
- Ladung dll erfolgt sofort nach der Ladung von Experten (oder eines anderen mql5-Programms);
- Prüfung der logischen Bedingungen ist verkürzt;
- die laufende Durchführung wird bei der Grenzenüberschreitung des Feldes gestoppt (kritisch - mit Fehlerausgabe) ;
- Priorität der Operationen wie in C ++;
- Explizite Typenreduzierung (auch von der Zeile in die Zahl);
- Lokale Variablen werden automatisch nicht initialisiert (außer Zeilen);
- Normale lokale Felder werden automatisch entfernt.

## Sonderfunktionen `init`, `start` und `deinit`

In der Sprache MQL4 gab es nur drei vorbestimmte Funktionen, die im Code eines Indikators, Ratgebers oder Scripts sein konnten (nicht beachtet Include-Dateien \*.mqh und Bibliothekdateien). Diese Funktionen fehlen in MQL5, aber ihre Analoge gibt es. In der Tabelle ist die ungefähre Entsprechung dieser Funktionen dargestellt.

| MQL4                | MQL5                  |
|---------------------|-----------------------|
| <code>init</code>   | <code>OnInit</code>   |
| <code>start</code>  | <code>OnStart</code>  |
| <code>deinit</code> | <code>OnDeinit</code> |

Funktionen [OnInit](#) und [OnDeinit](#) spielen dieselbe Rolle wie die Funktionen `init` und `deinit` in MQL4 - sie sind bestimmt für Positionieren des Codes, der bei der Initialisierung und bei der Deinitialisierung des mql5-Programms durchgeführt werden muss. Sie können diese Funktionen entsprechenderweise umbenennen oder lassen sie wie sie sind, aber in entsprechenden Stellen den Aufruf zuzufügen.

**Beispiel:**



```

void OnInit()
{
    //--- Funktion wird bei der Initialisierung aufgerufen
    init();
}
void OnDeinit(const int reason)
{
    //--- Funktion wird bei der Deinitialisierung aufgerufen
    deinit();
    //---
}

```

Funktion start wird durch die Funktion [OnStart](#) nur in Scripts ersetzt, für Experten und Indikator muss sie in [OnTick](#) und [OnCalculate](#) umbenennen. Der Code, der innerhalb des mql5-Programms durchzuführen ist, muss in diesen drei Funktionen sein :

| mql5-Programm             | Hauptfunktion |
|---------------------------|---------------|
| <a href="#">Script</a>    | OnStart       |
| <a href="#">Indikator</a> | OnCalculate   |
| <a href="#">Expert</a>    | OnTick        |

Wenn die Hautfunktion im Code des Indikators oder Scripts fehlt oder sich der Name dieser Funktion von dem erforderlichen Namen unterscheidet, wird diese Funktion nicht aufgerufen. D.H. wenn es im Ausgangskode des Scripts die Funktion OnStart fehlt, wird dieser Code als Ratgeber compilert.

Wenn die Funktion OnCalculate im Code des Indikators fehlt, ist die Compilierung dieses Indikators unmöglich.

## Vorbestimmte Variablen

In MQL5 gibt es keine solche Variablen wie Ask, Bid, Bars. Die Schreibweise der Variablen Digits und Point ist etwas verändert, wie es in der Tabelle gezeigt ist.

| MQL4   | MQL5          |
|--------|---------------|
| Digits | _Digits       |
| Point  | _Point        |
|        | _LastError    |
|        | _Period       |
|        | _Symbol       |
|        | _StopFlag     |
|        | _UninitReason |

## Zugang zu Zeitreihen

In MQL5 gibt es keine vorbestimmte Zeitreihen `Open[]`, `High[]`, `Low[]`, `Close[]`, `Volume[]` und `Time[]`. Die notwendige Tiefe der Zeitreihe kann selbstständig durch die entsprechenden [Funktionen für Zugang zu Zeitreihen](#) vorgegeben werden.

## Ratgeber (Experten)

Experten in MQL5 brauchen nicht unbedingt die Funktion-Bearbeiter des [Ereignisses](#) haben, wenn neues Tick `OnTick` empfangen wird, wie es in MQL4 der Fall war (Funktion `start` in MQL4 wird beim Empfang neues Ticks aufgerufen), denn jetzt können Experten in MQL5 vorbestimmte Funktionen-Bearbeiter mehrerer Typen der Ereignisse haben:

- [OnTick](#) - Empfang eines neuen Ticks;
- [OnTimer](#) - Ereignis des Timers;
- [OnTrade](#) - Handelsereignis;
- [OnChartEvent](#) - Ereignis der Eingabe von Keyboard und Maus, Ereignis der Verschiebung des graphischen Objektes, Ereignis der Beendigung der Textbearbeitung im Feld der Objektausgabe;
- [OnBookEvent](#) - Ereignis der DOM Veränderung (Depth of Market).

## Benutzerindikatoren

In MQL4 ist die Anzahl von Indikatorpuffer begrenzt und kann nicht mehr als 8 sein. In MQL5 gibt es keine solche Einschränkung, aber man muss sich daran erinnern, dass jeder Indikatorpuffer ein bestimmtes Volumen des Operativspeichers erfordert, darum muss man die neue Möglichkeit nicht missbrauchen.

Außerdem gab es in MQL4 nur 6 Typen der Zeichnung des Benutzerindikators, in MQL5 gibt es nun 18 [Zeichnungsstile](#). Obwohl sich die Namen der Zeichnungstypen nicht verändert haben, hat sich die Ideologie der graphischen Darstellung der Indikatoren wesentlich verändert.

Richtung des Indizierens in Indikatorpuffern unterscheidet sich auch. In MQL5 haben alle Puffer dasselbe Verhalten, wie normale Felder, d.h. Element mit dem Index 0 ist das älteste in der Geschichte, beim Steigen des Indexes bewegen wir von den ältesten Daten zu den neuesten.

Die einzige Funktion für die Arbeit mit [Benutzerindikatoren](#), die sich aus MQL4 erhält ist [SetIndexBuffer](#). Aber ihr Aufruf hat sich verändert, nun muss man [Typ der Daten, die im Feld aufbewahren werden](#) angeben, verbunden mit dem Indikatorpuffer.

Außerdem haben sich die Eigenschaften der Benutzerindikatoren verändert und erweitert, Funktionen für [Zugang zu Zeitreihen](#) erschienen, darum muss der ganze Algorithmus der Berechnung vollständig neu überlegt werden.

## Graphische Objekte

Die Anzahl der graphischen Objekte in MQL5 ist wesentlich gestiegen. Außerdem ist Positionieren der graphischen Objekte in der Zeit mit der Genauigkeit bis auf die Sekunde ist auf dem Chart jeder Periode möglich geworden - jetzt werden Bezugspunkte von graphischen Objekten mit der Genauigkeit bis auf die Eröffnungszeit der Bar im laufenden Preischart nicht ausgerundet.

für Objekte `Arrow`, `Text` und `Label` ist es möglich geworden, [Weg der Verbindung](#) anzugeben, und für Objekte `Label`, `Button`, `Chart`, `Bitmap Label` und `Edit` ist es möglich, [Chartwinkel, zu dem Objekt](#)

gebunden ist vorzugeben .